

New Results on Online Resource Minimization*

Lin Chen[†]

Nicole Megow[†]

Kevin Schewior[‡]

December 9, 2015

Abstract

We consider the online resource minimization problem in which jobs with hard deadlines arrive online over time at their release dates. The task is to determine a feasible schedule on a minimum number of machines.

We rigorously study this problem and derive various algorithms with small constant competitive ratios for interesting restricted problem variants. As the most important special case, we consider scheduling jobs with agreeable deadlines. We provide the first constant-ratio competitive algorithm for the non-preemptive setting, which is of particular interest with regard to the known strong lower bound of n for the general problem. For the preemptive setting, we show that the natural algorithm LLF achieves a constant ratio for agreeable jobs, while in general it has a lower bound of $\Omega(n^{1/3})$.

We also give an $\mathcal{O}(\log n)$ -competitive algorithm for the general preemptive problem, which improves upon a known $\mathcal{O}(\log(p_{\max}/p_{\min}))$ -competitive algorithm. Our algorithm maintains a dynamic partition of the job set into loose and tight jobs and schedules each (temporal) subset individually on separate sets of machines. The key is a characterization of how the decrease in the relative laxity of jobs influences the optimum number of machines. To achieve this we derive a compact expression of the optimum value, which might be of independent interest. We complement the general algorithmic result by showing lower bounds that rule out that other known algorithms may yield a similar performance guarantee.

*This research was supported by the German Science Foundation (DFG) under contract ME 3825/1. The third author was supported by the DFG within the research training group ‘Methods for Discrete Structures’ (GRK 1408).

[†]Technische Universität München, Zentrum für Mathematik, Garching, Germany. Email: nmegow@ma.tum.de, chenlin198662@gmail.com.

[‡]Technische Universität Berlin, Institut für Mathematik, Berlin, Germany. Email: schewior@math.tu-berlin.de.

1 Introduction

Minimizing the resource usage is a key to the achievement of economic, environmental or societal goals. We consider the fundamental problem of minimizing the number of machines that is necessary for feasibly scheduling jobs with release dates and hard deadlines. We consider the online variant of this problem in which every job becomes known to the online algorithm only at its release date. We denote this problem as *online machine minimization problem*. We also consider a *semi-online* variant in which we give the online algorithm slightly more information by giving the optimal number of machines as part of the input.

We develop several algorithms for preemptive and non-preemptive problem variants, and evaluate their performance by the competitive ratio, a widely-used measure that compares the solution value of an online algorithm with an optimal offline value. We derive the first constant-competitive algorithms for certain variants of (semi-)online machine minimization and improve several others. As a major special case, we consider online deadline scheduling with *agreeable deadlines*, i.e., when the order of deadlines for all jobs coincides with the order of their release dates. We give the first constant-competitive algorithm for scheduling agreeable jobs non-preemptively, which contrasts to the strong lower bound of n for the general problem [18]. We also prove that for the preemptive scheduling problem, one of the most natural algorithms called *Least Laxity First* (LLF) achieves a constant ratio for agreeable jobs, in contrast with its lower bound of $\Omega(n^{1/3})$ on the general problem. Our most general result is a $\mathcal{O}(\log n)$ -competitive algorithm for the preemptive scheduling problem.

Previous results. The preemptive semi-online machine minimization problem, in which the optimal number of machines is known in advance, has been investigated extensively by Phillips et al. [17], and there have hardly been any improvements since then. Phillips et al. show a general lower bound of $\frac{5}{4}$ and leave a huge gap to the upper bound $\mathcal{O}(\log \frac{p_{\max}}{p_{\min}})$ on the competitive ratio for the so-called *Least Laxity First* (LLF) Algorithm. Not so surprisingly, they also rule out that the *Earliest Deadline First* (EDF) Algorithm could improve on the performance of LLF; indeed they show a lower bound of $\Omega(\frac{p_{\max}}{p_{\min}})$. It is a wide open question if preemptive (semi-)online machine minimization admits a constant-factor online algorithm.

The non-preemptive problem is considerably harder than the preemptive problem. If the set of jobs arrives online over time, then no algorithm can achieve a constant or even sublinear competitive ratio [18]. However, relevant special cases admit online algorithms with small constant worst-case guarantees. The problem with unit processing times was studied in a series of papers [9,13,14,18,19] and implicitly in the context of energy-minimization in [4]. It has been shown that an optimal online algorithm has the exact competitive ratio $e \approx 2.72$ [4,9]. For non-preemptive scheduling of jobs with equal deadlines, an upper bound of 16 is given in [9]. We are not aware of any previous work on online machine minimization restricted to instances with agreeable deadlines. However, in other contexts, e.g., online buffer management [12] and scheduling with power management [1,3], it has been studied as an important and relevant class of instances.

Online deadline scheduling with extra speedup is another related problem variant. We are given the same input as in the semi-online machine minimization problem. Instead of equipping an online algorithm with additional machines, the given m machines may run at an increased speed $s \geq 1$. The goal is to find an online scheduling algorithm that requires minimum speed s . This problem seems much better understood and (nearly) tight speedup factors below 2 are known (see [2,15,17]). However, the power of speed is much stronger than additional machines since it allows parallel processing of jobs to some extent. Algorithms that perform well for the speed-problem, such as, EDF, LLF and a deadline-ordered algorithm, do not admit constant performance guarantees for the

	preemptive				non-preemptive			
	OPT known		OPT unknown		OPT known		OPT unknown	
	LB	UB	LB	UB	LB	UB	LB	UB
general	$\frac{5}{4}$ [17]	$\mathcal{O}(\log n)$	e [4, 9]	$\mathcal{O}(\log n)^*$	n [18]	—	n [18]	—
$p_j \leq \alpha(d_j - r_j)$	—	$\frac{1}{(1-\alpha)^2}$	—	$\frac{4}{(1-\alpha)^2}^*$	n [18]	—	n [18]	—
agreeable deadlines	—	18	e [4, 9]	72*	—	9	e [4, 9]	16
$d_j \equiv d$	—	1	e [4, 9]	4*	—	$5\frac{1}{4}$	e [4, 9]	$11\frac{1}{9}$
$p_j \equiv p$	$\frac{8}{7}$	3	e [4, 9]	9.38	$\frac{8}{7}$	4	e [4, 9]	10
$p_j \equiv 1$	—	1 [13]	e [4, 9]	e [4, 9]	—	1 [13]	e [4, 9]	e [4, 9]

Table 1: State of the art for (semi-)online machine minimization. New results presented in this paper are colored red (and have no reference). Results for the online setting that are directly derived from the semi-online setting are marked with a star (*). Trivial bounds are excluded (—).

machine minimization problem.

We also mention that the offline problem, in which all jobs are known in advance, can be solved optimally in polynomial time if job preemption is allowed [11]. Again, the problem complexity increases drastically if preemption is not allowed. In fact, the problem of deciding whether one machine suffices to schedule all the jobs non-preemptively is strongly NP-complete [10]. It is even open if a constant-factor approximation exists; a lower bound of $2 - \varepsilon$ was given in [8]. The currently best known non-preemptive approximation algorithm is by Chuzhoy et al. [7] who propose a sophisticated rounding procedure for a linear programming relaxation that yields an approximation factor of $\mathcal{O}(\sqrt{\frac{\log n}{\log \log n}})$. Small constant factors were obtained for special cases [8, 20]: in particular, Yu and Zhang [20] give a 2-approximation when all release dates are equal and a 6-approximation when all processing times are equal.

Our contribution. We develop several algorithms and bounding techniques for the (semi-) online machine minimization problem. We give an improved algorithm for the general problem and present a rigorous study of interesting restricted variants for which we show small constant competitive ratios. A summary of our new results can be found in Table 1.

As an important special case, we consider the class of instances with *agreeable deadlines* in which the order of deadlines for all jobs coincides with the order of their release dates. We give the first constant competitive ratio for the preemptive and non-preemptive problem variants. The constant performance guarantee for the non-preemptive setting is particularly interesting with regard to the strong lower bound of n for instances without this restriction [18]. For the preemptive setting, we show that the natural algorithm LLF admits a constant performance guarantee on agreeable jobs, in contrast with its non-constant ratio on the general problem. We also quantify how the tightness of jobs influences the performance of algorithms such as EDF and LLF, which might be of independent interest.

Our most general result is a $\mathcal{O}(\log n)$ -competitive algorithm for the unrestricted preemptive online problem. This improves on the $\mathcal{O}(\log(p_{\max}/p_{\min}))$ -competitive algorithm by Phillips et al. [16, 17], which was actually obtained for the semi-online problem in which the optimal number of machines is known in advance. But as we shall see it can be used to solve the online problem as well. In fact, we show that the online machine minimization problem can be reduced to the semi-online problem by losing at most a factor 4 in the competitive ratio. This is true for preemptive as well as non-preemptive scheduling.

Our algorithm for the general problem maintains a dynamic partition of jobs into loose and tight jobs and schedules each (temporal) subset individually on separate machines. The loose jobs are scheduled by EDF whereas the tight jobs require a more sophisticated treatment. The key is a characterization of how the decrease in the relative laxity of jobs influences the optimum number of machines. To quantify this we derive a compact expression of the optimum value, which might be of independent interest. We complement this algorithmic result by showing strong lower bounds on LLF or any deadline-ordered algorithm.

We remark that our algorithms run in polynomial time (except for the non-preemptive online algorithms). As a side result, we even improve upon a previous approximation result for non-preemptive offline machine minimization with jobs of equal processing time when the optimum is not known. Our semi-online 4-competitive algorithm can be easily turned into an offline 4-approximation which improves upon a known 6-approximation [20].

Outline. In Section 2, we introduce some notations and state basic results used throughout the paper. In Section 3, we give constant-competitive algorithms for the problem of scheduling agreeable jobs. We continue with the other special cases of equal processing times and uniform deadlines in Sections 4 and 5, respectively. Our general $\mathcal{O}(\log n)$ -competitive algorithm is presented in Section 6. We finally present some lower bounds in Section 7.

2 Problem Definition and Preliminaries

Problem definition. Given is a set of jobs $J = \{1, 2, \dots, n\}$ where each job $j \in J$ has a processing time $p_j \in \mathbb{N}$, a release date $r_j \in \mathbb{N}$ which is the earliest possible time at which the job can be processed, and a deadline $d_j \in \mathbb{N}$ by which it must be completed. The task is to open a minimum number of machines such that there is a feasible schedule in which no job misses its deadline. In a feasible schedule each job $j \in J$ is scheduled for p_j units of time within the time window $[r_j, d_j]$. Each opened machine can process at most one job at the time, and no job is running on multiple machines at the same time. When we allow job preemption, then a job can be preempted at any moment in time and may resume processing later on the same or any other machine. When preemption is not allowed, then a job must run until completion once it has started.

To evaluate the performance of our online algorithms we perform a *competitive analysis* (see e.g. [5]). We call an online algorithm A c -competitive if m_A machines with $m_A \leq c \cdot m$ suffice to guarantee a feasible solution for any instance that admits a feasible schedule on m machines.

Notation. We let $J(t) := \{j \in J \mid r_j \leq t\}$ denote the set of all jobs that have been released by time t . For some job set J we let $m(J)$ denote the minimum number of machines needed to feasibly schedule J . If J is clear from the context, we also write more concisely m for $m(J)$ and $m(t)$ for $m(J(t))$.

Consider an algorithm A and the schedule $A(J)$ it obtains for J . We denote the *remaining processing time* of a job j at time t by $p_j^{A(J)}(t)$. Further, we define the *laxity* of j at time t as $\ell_j^{A(J)}(t) = d_j - t - p_j^{A(J)}(t)$ and call $\ell_j = \ell_j(r_j)$ the *original laxity*. We denote the largest deadline in J by $d_{\max}(J) = \max_{j \in J} d_j$. We classify jobs by the fraction that their processing time takes from the feasible time window and call a job j α -loose at t if $p_j(t) \leq \alpha(d_j - r_j)$ or α -tight at t , otherwise. Here, we drop t if $t = r_j$. Finally, a job is called *active* at any time it is released but unfinished.

We analyze our algorithms by estimating the total processing volume (or workload) that is assigned to certain time intervals $[t, t']$. To this end, we denote by $w_{A(J)}(t)$ the total processing volume of J that A assigns to $[t, t+1)$ (or simply to t). Since our algorithms make decisions at

integral time points, this is simply the number of assigned jobs. Further, $W_{A(J)}(t)$ denotes the total remaining processing time of all unfinished jobs (including those not yet released). We omit the superscripts whenever J or A are non-ambiguous, and we use **OPT** to indicate an optimal schedule.

Reduction to the semi-online problem. In the following we show that we may assume that the optimum number of machines m is given by losing at most a factor 4 in the competitive ratio.

We employ the general idea of *doubling* an unknown parameter, which has been successfully employed in solving various online optimization problems [6]. In our case, the unknown parameter is the optimal number of machines. The idea is to open additional machines at any time that the optimum solution has doubled.

Let $A_\alpha(m)$ denote an α -competitive algorithm for the semi-online machine minimization problem given the optimum number of machines m . Then our algorithm for the online problem is as follows.

Algorithm Double:

- Let $t_0 = \min_{j \in J} r_j$. For $i = 1, 2, \dots$ let $t_i = \min\{t \mid m(t) > 2m(t_{i-1})\}$.
- At any time t_i , $i = 0, 1, \dots$, open $2\alpha m(t_i)$ additional machines. All jobs with $r_j \in [t_{i-1}, t_i)$ are scheduled by Algorithm $A_\alpha(2m(t_{i-1}))$ on the machines opened at time t_{i-1} .

Observe that this procedure can be executed online, since the time points t_0, t_1, \dots as well as $m(t_0), m(t_1), \dots$ can be computed online and A_α is assumed to be an algorithm for the semi-online problem. Since the optimal solution increases only at release dates, which are integral, we may assume that $t_i \in \mathbb{N}$. Notice also that **Double** does not preempt jobs which would not have been preempted by Algorithm A_α .

Theorem 1. *Given an α -competitive algorithm for the (non)-preemptive semi-online machine minimization problem, Double is 4α -competitive for (non)-preemptive online machine minimization.*

Proof. Let t_0, t_1, \dots, t_k denotes the times at which **Double** opens new machines. **Double** schedules the jobs $J_i = \{j \mid r_j \in [t_i, t_{i+1})\}$, with $i = 0, 1, \dots, k$, using Algorithm $A_\alpha(2m(t_i))$ on $2\alpha m(t_i)$ machines exclusively. This yields a feasible schedule since an optimal solution for $J_i \subseteq J(t_{i+1} - 1)$ requires at most $m(t_{i+1} - 1) \leq 2m(t_i)$ machines and the α -competitive subroutine $A_\alpha(2m(t_i))$ is guaranteed to find a feasible solution given a factor α times more machines than optimal, i.e., $2\alpha m(t_i)$.

It remains to compare the number of machines opened by **Double** with the optimal number of machines m , which is at least $m(t_k)$. By construction it holds that $2m(t_i) \leq m(t_{i+1})$, which implies by recursive application that

$$2m(t_i) \leq \frac{m(t_k)}{2^{k-i-1}}. \quad (1)$$

The total number of machines opened by **Double** is

$$\sum_{i=0}^k 2\alpha m(t_i) \leq \sum_{i=0}^k \alpha \frac{1}{2^{k-i-1}} m(t_k) = 2\alpha \sum_{i=0}^k \frac{1}{2^i} m(t_k) \leq 4\alpha m,$$

which concludes the proof. \square

Using standard arguments, this performance guarantee can be improved to a factor $e \approx 2.72$ using randomization over the doubling factor. However, in the context of hard real-time guarantees a worst-case guarantee that is achieved in expectation seems less relevant and we omit it.

Algorithms. An algorithm for the (semi-)online machine minimization problem is called *busy* if, at all times t , $w_A(t) < m$ implies that there are exactly $w_A(t)$ active jobs at t . Two well-known busy algorithms are the following. The *Earliest Deadline First* (EDF) algorithm on m' machines schedules at any time the m' jobs with the smallest deadline. The *Least Laxity First* (LLF) algorithm on m' machines, schedules at any time the m' jobs with the smallest non-negative laxity at that time. In both cases, ties are broken in favor of earlier release dates or, in case of equal release dates, by lower job indices. For simplicity, we assume in this paper that both EDF and LLF make decisions only at integer time points.

Scheduling loose jobs. Throughout the paper, we will use the fact that, for any fixed $\alpha < 1$, we can achieve a constant competitive ratio on α -loose jobs, i.e., when $p_j(t) \leq \alpha(d_j - r_j)$, simply by using EDF. We remark that the same result also applies to LLF.

Theorem 2. *If every job is α -loose, EDF is a $1/(1 - \alpha)^2$ -competitive algorithm for preemptive semi-online machine minimization.*

We prove this theorem by establishing a contradiction based on the workload that EDF when missing deadlines must assign to some interval. We do so by using the following work load inequality, which holds for arbitrary busy algorithms.

Lemma 1. *Let every job be α -loose, $c \geq 1/(1 - \alpha)^2$ and let A be a busy algorithm for semi-online machine minimization using m machines. Assume that $\ell_j^A(t') \geq 0$ holds, for all $t' \leq t$ and j . Then*

$$W_A(t) \leq W_{\text{OPT}}(t) + \frac{\alpha}{1 - \alpha} \cdot m \cdot (d_{\max} - t).$$

Proof. We prove by induction. The base is clear since $W_A(0) = W_{\text{OPT}}(0)$. Now assume the lemma holds for all $t' < t$. There are three possibilities.

Case 1. We have $w_A(t) \leq \alpha/(1 - \alpha) \cdot m$. By the fact that A is busy, there is an empty machine at time t , meaning that there are at most $\alpha/(1 - \alpha) \cdot m$ active jobs. Given that $\ell_j^A(t') \geq 0$ for all $t' \leq t$, each of the active jobs has a remaining processing time of no more than $d_{\max} - t$, implying that $W_A(t)$ is bounded by $\alpha/(1 - \alpha) \cdot m \cdot (d_{\max} - t)$ plus the total processing time of the jobs that are not released. As the latter volume cannot exceed $W_{\text{OPT}}(t)$, the claim follows.

Case 2. We have $w_A(t) \geq 1/(1 - \alpha) \cdot m$. According to the induction hypothesis, it holds that $W_A(t - 1) \leq W_{\text{OPT}}(t - 1) + \alpha/(1 - \alpha) \cdot m \cdot (d_{\max} - t + 1)$. Using $w_{\text{OPT}}(t) \leq m$, we get $W_{\text{OPT}}(t) \geq W_{\text{OPT}}(t - 1) - m$. For Algorithm A , it holds that $W_A(t) = W_A(t - 1) - w_A(t) \leq W_A(t - 1) - 1/(1 - \alpha) \cdot m$. Inserting the first inequality into the second one proves the claim.

Case 3. We have $\alpha/(1 - \alpha) \cdot m < w_A(t) < 1/(1 - \alpha) \cdot m$. Again by the fact that A is busy, there is an empty machine at time t , i.e., there are less than $1/(1 - \alpha) \cdot m$ active jobs. Distinguish two cases.

Case 3a. We have $p_j(t) \leq \alpha(d_j - t)$ for all active jobs. Then the total remaining processing time of active jobs is bounded by $\alpha(d_{\max} - t) \cdot 1/(1 - \alpha) \cdot m$. Plugging in the total processing time of unreleased jobs, which is bounded by $W_{\text{OPT}}(t)$, we again get the claim.

Case 3b. There exists an active job j with $p_j(t) > \alpha(d_j - t)$. Using that j is α -loose, i.e., $p_j \leq \alpha(d_j - r_j)$, we get that j is not processed for at least $(1 - \alpha)(t - r_j)$ many time units in the interval $[r_j, t]$. As Algorithm A is busy, this means that all machines are occupied at these times,

yielding

$$\begin{aligned}
W_A(t) &\leq W_A(r_j) - (1 - \alpha)(t - r_j) \cdot cm \\
&\leq W_{\text{OPT}}(r_j) + \frac{\alpha}{1 - \alpha} \cdot m \cdot (d_{\max} - r_j) - (1 - \alpha)(t - r_j) \cdot cm \\
&\leq W_{\text{OPT}}(r_j) + \frac{\alpha}{1 - \alpha} \cdot m \cdot (d_{\max} - t) - m \cdot (t - r_j),
\end{aligned}$$

where the second inequality follows by the induction hypothesis for $t = r_j$, and the third one follows by $c \geq 1/(1 - \alpha)^2$. Lastly, the feasibility of the optimal schedule implies

$$W_{\text{OPT}}(t) \geq W_{\text{OPT}}(r_j) - m \cdot (t - r_j),$$

which in turn implies the claim by plugging it into the former inequality. \square

Proof of Theorem 2. Let cm be the number of machines EDF uses and consider the schedule produced by EDF from an arbitrary instance J only consisting of α -loose jobs. We have to prove that every job is finished before its deadline if $c = 1/(1 - \alpha)^2$. To this end, suppose that EDF fails. Among the jobs that are missed, let j^* be one of those with the earliest release date.

First observe that we can transform J into $J' \subset J$ such that $j^* \in J'$, $\max_{j \in J'} d_j = d_{j^*}$ and EDF still fails on J' . For this purpose, simply define $J' = \{j \in J \mid d_j \leq d_{j^*}\}$ and notice that we have only removed jobs with a later deadline than $\max_{j \in J'} d_j$. This implies that every job from J' receives the same processing in the EDF schedule of J' as in the one of J .

We can hence consider J' instead of J from now on. In the following, we establish a contradiction on the workload during the time interval $[r_{j^*}, d_{j^*}]$. The first step towards this is applying Lemma 1 for an upper bound. Also making use of the feasibility of the optimal schedule, we get:

$$\begin{aligned}
W_{\text{EDF}}(r_{j^*}) &\leq W_{\text{OPT}}(r_a) + \frac{\alpha}{1 - \alpha} \cdot m \cdot (d_{\max} - r_{j^*}) \\
&\leq \frac{1}{1 - \alpha} \cdot m \cdot (d_{\max} - r_{j^*}).
\end{aligned}$$

We can, however, also lower bound this workload. Thereto, note that job j^* is not processed for at least $(1 - \alpha)(d_{j^*} - r_{j^*})$ time units, implying that all machines must be occupied by then, i.e.,

$$W_{\text{EDF}}(r_{j^*}) > (1 - \alpha) \cdot cm \cdot (d_{\max} - r_{j^*}).$$

If we compare the right-hand sides of the two inequalities, we arrive at a contradiction if and only if $c \geq 1/(1 - \alpha)^2$, which concludes the proof. \square

3 Constant-competitive Algorithms for Agreeable Deadlines

In this section, we study the special case of (semi-)online machine minimization of jobs with *agreeable deadlines*. That is, for any two jobs $j, k \in J$ with $r_j < r_k$ it holds that also $d_j \leq d_k$.

3.1 Preemptive scheduling

We propose an algorithm that again schedules α -loose jobs by EDF or LLF on $m/(1 - \alpha)^2$ machines, which is feasible by Theorem 2. Furthermore, α -tight jobs are scheduled by LLF, for which we obtain the following bound.

Theorem 3. *The competitive ratio of LLF is at most $4/\alpha + 6$ if all jobs are α -tight and have agreeable deadlines.*

The proof is based on a load argument. Notice that Lemma 1 is no longer true since jobs now are all tight. To define a new load inequality we introduce the set $\Lambda(t)$ which is defined as follows. Consider the latest point in time before (or at) t such that at most m jobs are processed and denote it by $\phi(t) = \max\{t' \leq t \mid w_{\text{LLF}}(t') \leq m\}$. Let $\Lambda(t)$ be the set of active jobs at time $\phi(t)$. Obviously $|\Lambda(t)| \leq m$. If there does not exist such a $\phi(t)$, i.e., $w(x) > m$ holds for any $x \leq t$, then we let $\Lambda(t) = \emptyset$.

Lemma 2. *Consider an instance consisting only of α -tight jobs. Let $c \geq 4/\alpha + 6$ and t such that there is no job missed by LLF up to time t . Then we have*

$$W_{\text{LLF}}(t) \leq W_{\text{OPT}}(t) + \sum_{j \in \Lambda(t)} p_j(t).$$

Proof. The lemma is trivially true for $t = 0$. Now assume $t > 0$ and that the lemma holds for all $t' < t$. We make the following assumptions. Indeed, if any of them is not true, then we can easily prove the lemma.

- (i) $\phi(t) = t_l$ exists, and $t_l < t$.
- (ii) $\Lambda(t) \neq \emptyset$.
- (iii) During $[t_l, t]$, at least one job is not processed all the time.
- (iv) $d_j \geq t$ for at least one job $j \in \Lambda(t)$.

Consider assumption (i). The lemma is obviously true if $\phi(t)$ does not exist because that means $w_{\text{LLF}}(t') > m$ for any $t' \leq t$. That is at time t , LLF must have finished more load than the optimum solution, in which only m machines are available. It follows directly that $W_{\text{LLF}}(t) \leq W_{\text{OPT}}(t)$. Otherwise there exists $\phi(t) = t_l \leq t$. Notice that, if $t_l = t$, then obviously $W_{\text{LLF}}(t_l) \leq W_{\text{OPT}}(t_l) + \sum_{j \in \Lambda(t_l)} p_j(t_l)$. Thus we assume in the following that $t_l < t$, and it follows that $\Lambda(t') = \Lambda(t_l)$ for any $t' \in [t_l, t]$.

Consider assumption (ii). If $\phi(t) = t_l$ exists while $\Lambda(t)$ is empty, then LLF finishes all the jobs released before time t_l , while $w_{\text{LLF}}(t') > m \geq w_{\text{OPT}}(t')$ for $t_l < t' \leq t$. It is easy to see that the lemma follows.

Consider assumption (iii). Notice that $W_{\text{LLF}}(t) \leq W_{\text{OPT}}(t) + \sum_{j \in \Lambda(t)} p_j(t)$ is obviously true if no new job is released after t_l . If a new job is released after t_l and scheduled without preemption by LLF, then its contribution to $W_{\text{LLF}}(t)$ is no more than its contribution to $W_{\text{OPT}}(t)$. Plugging in the contributions of the newly released jobs, we still have $W_{\text{LLF}}(t) \leq W_{\text{OPT}}(t) + \sum_{j \in \Lambda(t)} p_j(t)$.

Consider assumption (iv). If it is not true, we have $W_{\text{LLF}}(t-1) \leq W_{\text{OPT}}(t-1)$ according to the induction hypothesis, and $p_j(t-1) = 0$ holds for all $j \in \Lambda(t-1)$. Given that $W_{\text{LLF}}(t) \leq W_{\text{LLF}}(t-1) - m$ and $W_{\text{OPT}}(t-1) \geq W_{\text{OPT}}(t) - m$, the lemma is true.

We proceed with the four assumptions above. Notice that some job is not processed in $[t_l, t]$, and we let $[t_0, t_0+1]$ be the first such time. We let t_a be the first time in $[t_0, t]$ with $w_{\text{LLF}}(t_a) \leq 2m$. Notice that t_a might not exist, and in this case $w_{\text{LLF}}(t') > 2m$ holds for all $t' \in [t_0, t]$. As a consequence we get $W_{\text{LLF}}(t) \leq W_{\text{LLF}}(t_0) - 2m \cdot (t - t_0)$. On the other hand, $W_{\text{OPT}}(t) \geq W_{\text{OPT}}(t_0) - m \cdot (t - t_0)$. Given the fact that $\sum_{j \in \Lambda(t)} p_j(t) \leq \sum_{j \in \Lambda(t)} p_j(t_0) - m \cdot (t - t_0)$, we have $W_{\text{LLF}}(t) \leq W_{\text{OPT}}(t) + \sum_{j \in \Lambda(t)} p_j(t)$.

Now assume that $t_a \leq t$ exists. Let $t_b \leq t_0$ be the latest time with $w_{\text{LLF}}(t_b - 1) \leq \gamma cm$ for some parameter $\gamma < 1$ that will be fixed later (i.e., $w(t') > \gamma cm$ for $t' \in [t_b, t_0]$). Given that $w_{\text{LLF}}(t_l) \leq m$, we know that $t_b \in [t_l, t_0] \subseteq [t_l, t]$, and furthermore that

$$W_{\text{LLF}}(t) \leq W_{\text{LLF}}(t_b) - \gamma cm \cdot (t_0 - t_b). \quad (2)$$

Since $w_{\text{LLF}}(t_b - 1) \leq \gamma cm$ and $w_{\text{LLF}}(t_0) = cm$, at least $(1 - \gamma) \cdot cm$ jobs are released during $[t_b, t_0]$. Since $w_{\text{LLF}}(t_a) \leq 2m$, among these $(1 - \gamma) \cdot cm$ jobs, there are at least $(1 - \gamma - 2/c) \cdot cm$ finished before t_a . Let j be any of these jobs released after t_b and finished before t_a . Now consider the active jobs at time t_l . We know by (iv) that at least one of them, say job j' , has a deadline no less than t . Recall that we are given an agreeable instance, hence job j that is released after j' must have a larger deadline, i.e., $d_j \geq t$. Further, we have $r_j \leq t_0$, and as job j is a tight job it follows that $p_j \geq \alpha \cdot (d_j - r_j) \geq \alpha \cdot (t - t_0)$. Thus,

$$W_{\text{LLF}}(t) \leq W_{\text{LLF}}(t_b) - (1 - \gamma - 2/c) \cdot \alpha cm \cdot (t - t_0). \quad (3)$$

By combining Inequalities 2 and 3, we have

$$\begin{aligned} W_{\text{LLF}}(t) &\leq W_{\text{LLF}}(t_b) - \max\{\gamma \cdot cm \cdot (t_0 - t_b), (1 - \gamma - 2/c) \cdot \alpha cm \cdot (t - t_0)\} \\ &\leq W_{\text{LLF}}(t_b) - 1/2 \cdot \gamma cm \cdot (t - t_b). \end{aligned} \quad (4)$$

if we choose γ such that $\alpha \cdot (1 - \gamma - 2/c) = \gamma$.

On the other hand, it is easy to observe the following three inequalities: $W_{\text{LLF}}(t_b) \leq W_{\text{OPT}}(t_b) + \sum_{j \in \Lambda(t)} p_j(t_b)$, $W_{\text{OPT}}(t) \geq W_{\text{OPT}}(t_b) - m \cdot (t - t_b)$, and $\sum_{j \in \Lambda(t)} p_j(t) \geq \sum_{j \in \Lambda(t)} p_j(t_b) - m \cdot (t - t_b)$. Hence it follows that

$$W_{\text{LLF}}(t_b) \leq W_{\text{OPT}}(t) + m \cdot (t - t_b) + \sum_{j \in \Lambda(t)} p_j(t) + m \cdot (t - t_b). \quad (5)$$

By combining Inequalities 4 and 5, it can be easily seen that the lemma holds if we set $\gamma/2 \cdot c \geq 2$. Combining this inequality with the equation $\alpha \cdot (1 - \gamma - 2/c) = \gamma$ above, we get $c \geq 4/\alpha + 6$, i.e., for $c \geq 4/\alpha + 6$, the lemma is true. \square

Proof of Theorem 3. Assume the contrary, i.e., LLF fails. From the counterexamples, we pick one with the minimum number of jobs and let J denote the instance. Among all those jobs that are missed, we let z be the job with the earliest release date. Since job z is missed, there must exist some time t_0 when the laxity of job z is 0 but it is still not processed in favor of cm other jobs. Let these jobs be job 1 to job cm with $r_1 \leq r_2 \leq \dots \leq r_{cm}$ and S_z be the set of them. We have the following observations used throughout the proof:

- No job is released after t_0 .
- For $1 \leq i \leq cm$, we have $d_i \leq d_z$.
- For $1 \leq i \leq cm$, each job i has 0 laxity at t_0 and is not missed. Hence it is processed without preemption until time d_i .

Let $t_b = r_{\lfloor cm/3 \rfloor}$ and $t_a = d_{\lfloor 2cm/3 \rfloor}$. Consider time t_b . According to Lemma 2, we have

$$W_{\text{LLF}}(t_b) \leq W_{\text{OPT}}(t_b) + \sum_{j \in \Lambda(t_b)} p_j(t_b) \leq W_{\text{OPT}}(t_b) + m \cdot (t_a - t_b). \quad (6)$$

To see why the second inequality holds, observe that there are $\lfloor cm/3 \rfloor > m$ jobs being processed at time t_b . Thus any job in $\Lambda(t_b)$ must have a release date no later than r_m and hence it has a deadline no later than $d_m \leq t_a$. Furthermore, no job from $\Lambda(t_b)$ is missed, so the total remaining load of these jobs at t_a is at most $m \cdot (t_a - t_b)$.

We focus on the interval $[t_b, t_a]$ and will establish a contradiction on the load processed by LLF during this interval. We define a job j to be a *bad* job if it contributes more to $W_{\text{OPT}}(t_a)$ than to $W_{\text{LLF}}(t_a)$, i.e., either one of the following is true:

- Job j is not finished at t_a in both LLF and OPT, and we have $p_j^{\text{OPT}}(t_a) > p_j^{\text{LLF}}(t_a)$.
- At t_a , job j is finished in LLF but it is not finished in OPT.

Observation 1. *If the laxity of a job becomes 0 in LLF at some time during $[t_b, t_a]$, then it is not a bad job.*

Hence, any job in S_z cannot be a bad job since their laxities are 0 at $t_0 \in [t_b, t_a]$. Since we have $t_a = d_{\lfloor 2/3cm \rfloor}$, and a bad job j is not finished in OPT at time t_a , we get that $d_j \geq d_{\lfloor 2/3cm \rfloor}$, and $r_j \geq r_{\lfloor 2/3cm \rfloor} \geq t_b = r_{\lfloor cm/3 \rfloor}$. Now delete all the bad jobs in the schedules produced by LLF and OPT and denote their corresponding remaining loads at some t by $\hat{W}_{\text{LLF}}(t)$ and $\hat{W}_{\text{OPT}}(t)$, respectively. By the preceding observations, we again have

$$\hat{W}_{\text{LLF}}(t_b) \leq \hat{W}_{\text{OPT}}(t_b) + m \cdot (t_a - t_b)$$

because the same amount is subtracted from both sides of Inequality 6. Further, by the definition of bad jobs, we have

$$\hat{W}_{\text{LLF}}(t_a) \geq \hat{W}_{\text{OPT}}(t_a).$$

It follows directly that

$$\hat{W}_{\text{LLF}}(t_b) - \hat{W}_{\text{LLF}}(t_a) \leq 2m \cdot (t_a - t_b),$$

and we show in the following that

$$\hat{W}_{\text{LLF}}(t_b) - \hat{W}_{\text{LLF}}(t_a) \geq \lfloor cm/3 \rfloor \cdot (t_a - t_b) > 2m \cdot (t_a - t_b),$$

which will yield a contradiction and thus prove the theorem. Towards proving the claim, let $\hat{w}_{\text{LLF}}(t)$ be the load of LLF at time t in the schedule of LLF after deleting bad jobs. We prove that for any $t \in [t_b, t_a]$, we have $\hat{w}_{\text{LLF}}(t) \geq \lfloor cm/3 \rfloor$.

We first observe that at any $t \in [t_0, t_a]$ job $\lfloor 2cm/3 \rfloor + 1$ to job cm of S_z are processed. Thus the statement is true for every $t \in [t_0, t_a]$.

Consider any time $t \in [t_b, t_0]$. The first $\lfloor cm/3 \rfloor$ jobs from S_z are released and not finished, which implies that $w_{\text{LLF}}(t) \geq \lfloor cm/3 \rfloor$. Obviously if at time t no bad job is processed, then $\hat{w}_{\text{LLF}}(t) = w_{\text{LLF}}(t)$ and we are done. Otherwise at time t some bad job is being processed. We claim that, during $[t_b, t_0]$, the jobs $1, \dots, \lfloor cm/3 \rfloor$ from S_z will never be preempted in favor of a bad job, and then $\hat{w}_{\text{LLF}}(t) \geq \lfloor cm/3 \rfloor$ follows. Suppose this is not true. Then there exists some time $t \in [t_b, t_0]$, some bad job j and some other job $j' \in \{1, 2, \dots, \lfloor cm/3 \rfloor\}$ such that j' is preempted at t while j is being processed. We focus on the two jobs j and j' , and pick up the last time t' such that j' is preempted while j is being processed, and obviously $t' < t_0$. Since j' is preempted at t' , we get $\ell_j(t') \leq \ell_{j'}(t')$. We know that the laxity of a job decreases by 1 if it is preempted, and remains the same if it gets processed. From time $t' + 1$ to t_0 , we know that j' will never be preempted in favor of j since t' is the last such time. This means whenever $\ell_{j'}(t)$ decreases by 1 (meaning that j' is preempted), then $\ell_j(t)$ also decreases by 1. So from $t' + 1$ to t_0 , $\ell_{j'}(t)$ decreases no more than $\ell_j(t)$, implying that $\ell_j(t_0) \leq \ell_{j'}(t_0) = 0$, which contradicts the fact that j is a bad job. \square

Recall that α -loose jobs can be scheduled via EDF or LLF on $m/(1 - \alpha)^2$ machines. If we set $\alpha = 1/2$ and use LLF on the α -tight jobs, we get the following result.

Theorem 4. *If deadlines are agreeable, there is a 18-competitive algorithm for preemptive semi-online machine minimization.*

By applying Double, we get the following result for the fully online case.

Theorem 5. *If deadlines are agreeable, there is a 72-competitive algorithm for preemptive online machine minimization.*

3.2 Non-preemptive Scheduling

Consider the non-preemptive version of machine minimization with agreeable deadlines. Again we schedule α -tight and α -loose jobs separately. In particular, α -loose jobs are scheduled by a non-preemptive variant of EDF, that is, at any time $[t, t + 1]$, jobs processed at $[t - 1, t]$ and are not finished continue to be processed. If there are free machines, then among all the jobs that are not processed we select the ones with earliest deadlines and process them. To schedule α -tight jobs, we use MediumFit:

Algorithm MediumFit: Upon the release of a job j , we schedule it non-preemptively during the medium part of its processing interval, i.e., $[r_j + 1/2 \cdot \ell_j, d_j - 1/2 \cdot \ell_j]$.

We first prove a lemma about EDF and the α -tight jobs.

Lemma 3. *Non-preemptive EDF is a $1/(1 - \alpha)^2$ -competitive algorithm for non-preemptive semi-online machine minimization problem if all jobs are α -loose and have agreeable deadlines.*

Proof. We use the same technique as Theorem 2, i.e., we prove via contradiction on the workload.

We observe that the load inequality provided by Lemma 1 is also true for non-preemptive machine minimization. Suppose EDF fails, then among all the jobs that are missed we consider the job with the earliest release date and let it be j^* . Again let $J' = \{j \in J \mid d_j \leq d_{j^*}\}$. We claim that EDF also fails on J' . To see why, notice that jobs in $J \setminus J'$ have a larger deadline than j^* . Hence according to the agreeable deadlines their release dates are larger or equal to r_{j^*} . Furthermore according to EDF they are of a lower priority in scheduling, thus the scheduling of job j^* could not be delayed in favor of jobs in $J \setminus J'$. This implies that EDF also fails on J' . The remaining argument is the same as the proof of Theorem 2. \square

We remark that for non-agreeable deadlines, EDF is not a constant-competitive algorithm for α -loose jobs. Indeed, non-preemptive (semi-)online machine minimization does not admit algorithms using less than n machines when $m = 1$, even if every job is α -loose for any constant α [18]. The above argument fails as we can no longer assume that the job j^* missed by EDF has the largest deadline.

We continue with a lemma about MediumFit:

Lemma 4. *MediumFit is a $(2\lceil 1/\alpha \rceil + 1)$ -competitive (semi-)online algorithm for non-preemptive machine minimization on α -tight jobs with agreeable deadlines.*

Proof. Suppose that the algorithm runs more than $[2\lceil 1/\alpha \rceil + 1] \cdot m$ jobs at some time t . By the generalized pigeonhole principle, $2\lceil 1/\alpha \rceil + 2$ of them must run on the same machine in OPT. Consequently, on this machine, $\lceil 1/\alpha \rceil + 1$ of them are not finished after or not started before t . W.l.o.g. assume that the latter is the case and call these jobs J^* (the other case is symmetric).

Observe that, as J^* has been completely released at t , we can assume that, in OPT , these jobs are scheduled in order of their release dates, for this is the order of their deadlines. Now choose j^* with maximum release date, that is, also maximum deadline, from J^* . It holds that

$$t + \sum_{j \in J^*} p_j \leq d_{j^*} = r_{j^*} + \ell_{j^*} + p_{j^*}.$$

Plugging in $r_{j^*} \leq t - \ell_{j^*}/2$, which follows from the definition of **MediumFit**, yields

$$\sum_{j \in J^* \setminus \{j^*\}} p_j \leq \frac{\ell_{j^*}}{2}.$$

Hence, for all $j \in J^* \setminus \{j^*\}$, we have

$$r_j \leq r_{j^*} \leq t - \frac{\ell_{j^*}}{2} \leq t - \sum_{j' \in J^* \setminus \{j^*\}} p_{j'},$$

which follows by our choice of j^* as well as the previous inequalities. Since all of these jobs are α -tight, we have

$$p_j \geq \alpha \cdot (t - r_j) \geq \alpha \cdot \sum_{j' \in J^* \setminus \{j^*\}} p_{j'}.$$

By summing over all such j , we get

$$\begin{aligned} \sum_{j \in J^* \setminus \{j^*\}} p_j &> \alpha \cdot (|J^*| - 1) \cdot \sum_{j \in J^* \setminus \{j^*\}} p_j \\ &\geq \alpha \cdot \left\lceil \frac{1}{\alpha} \right\rceil \cdot \sum_{j \in J^* \setminus \{j^*\}} p_j \geq \sum_{j \in J^* \setminus \{j^*\}} p_j, \end{aligned}$$

which is a contradiction. \square

Combining the above two lemmas and choosing $\alpha = 1/2$, we have a guarantee for the semi-online case.

Theorem 6. *For non-preemptive machine minimization with agreeable deadlines, there is a semi-online 9-competitive algorithm.*

A fully online $(2 \cdot \lceil 1/\alpha \rceil + 1 + 4/(1 - \alpha)^2)$ -competitive algorithm can be obtained by using **MediumFit** for tight jobs and combining EDF with **Double**. Setting $\alpha = 1/3$ we get the following.

Theorem 7. *For non-preemptive machine minimization with agreeable deadlines, there is a online 16-competitive algorithm.*

4 Equal processing times

Consider the special case of (semi-)online machine minimization of jobs that have equal processing times, that is, $p_j = p$ for all $j \in J$.

4.1 Preemptive Scheduling

We first study preemptive semi-online machine minimization on instances with equal processing times. We firstly show that EDF is 3-competitive for semi-online machine minimization. Then we complement this result by a lower bound of $8/7$ on the competitive ratio of any deterministic online algorithm for the special case of equal processing times. Finally, we give a 9.38-competitive online algorithm.

Theorem 8. *EDF is a 3-competitive algorithm for semi-online machine minimization when all processing times are equal.*

Proof. Suppose the theorem is not true. Among those instances EDF fails at, we pick one with the minimum number of jobs. It is easy to see that in this counterexample there is only one job missing its deadline. Let j be this job. Then d_j is the maximum deadline among all the jobs and we let $d_j = d$. Furthermore, during $[d-p, d]$ there must be some time when job j is preempted. Let $t_0 \in [d-p, d]$ be the time when job j is preempted, and S_{t_0} be the set of jobs that are processed at time t_0 .

We claim that the total workload processed by EDF during $[t_0-p, d]$ is at least $3mp$. To see why, consider the EDF schedule. For simplicity we assume that S_{t_0} contains job 1 to job $3m$ and job i is always scheduled on machine i during $[t_0-p, d]$ for $1 \leq i \leq 3m$. We show that the workload of machine i in $[t_0-p, d]$ is at least p and the claim follows. Consider job i . Then due to the fact that job i does not miss its deadline, we know that the workload on machine i during $[t_0, d]$ is at least $p_i(t_0)$. Consider any time in $[t_0-(p-p_i(t_0)), t_0]$. either job i is scheduled or it is not in favor of other jobs. In both cases, the workload of machine i is at least $p-p_i(t_0)$. Thus, the workload of machine i during $[t_0-p, d]$ is at least p .

Since EDF fails, the previous considerations yield

$$W_{\text{EDF}}(t_0-p) > 3mp. \quad (7)$$

By the feasibility of the instance however, we know that

$$W_{\text{OPT}}(t_0-p) \leq m \cdot [d - (t_0-p)] \leq 2mp \quad (8)$$

(recall that $t_0 \geq d-p$). Therefore before time t_0-p , there must exist some time t when $w_{\text{EDF}}(t) \leq w_{\text{OPT}}(t) \leq m$ and, if t' is the latest such time, that $w_{\text{EDF}}(t) > w_{\text{OPT}}(t)$ for any $t' < t < t_0$. As there are at most m active jobs at time t' in the EDF schedule, we now get that

$$W_{\text{EDF}}(t') \leq W_{\text{OPT}}(t') + mp. \quad (9)$$

On the other hand, we have the inequalities $W_{\text{EDF}}(t') \geq W_{\text{EDF}}(t_0-p) + m \cdot (t_0-p-t')$ and $W_{\text{OPT}}(t') \leq W_{\text{OPT}}(t_0-p) + m \cdot (t_0-p-t')$. Plugging in Inequalities 7 and 8, respectively, into both of the latter ones yields a contradiction to Inequality 9. \square

Moreover, we provide a non-trivial lower bound. The underlying structure resembles the one known from the lower bound proof for the general semi-online machine minimization problem from [17].

Theorem 9. *Let $c < 8/7$. On instances fulfilling $p_j \equiv p \in \mathbb{N}$, there does not exist a c -competitive algorithm for the semi-online machine minimization problem.*

W.l.o.g. we restrict to even m . We will heavily make use of the following technical lemma.

Lemma 5. *Let $c < 8/7$, m be even and A be a c -competitive algorithm for semi-online machine minimization.*

Assume that J with $p_j = 2$, for all $j \in J$, is an instance with the following properties:

- (i) $W_{\text{OPT}(J)}(t) = 0$,
- (ii) $W_{A(J)}(t) \geq w$ and
- (iii) $d_j = t + 3$ for all jobs j active at t in $A(J)$.

Then there exists an instance $J' \supset J$ such that $p_j = 2$, for all $j \in J'$, with the following properties:

- (i) $W_{\text{OPT}(J')}(t + 3) = 0$,
- (ii) $W_{A(J')}(t + 3) \geq w' > w$ and
- (iii) $d_j = t + 6$ for all jobs j active at $t + 3$ in $A(J')$.

Proof. We augment J the following way. At t , we release m more tight jobs and $m/2$ more loose jobs. For each job j of the more tight jobs, we set $r_j = t$ and $d_j = t + 3$; for each of the more loose jobs, we set $r_j = t$ and $d_j = t + 6$.

We first observe that any c -competitive algorithm A has to do at least $w + 2m - 2m \cdot (c - 1) = w + 2m \cdot (2 - c)$ work on the remaining active jobs from J as well as the more tight jobs (i.e., all active jobs except for the more loose ones) in the interval $[t, t + 2]$. That is because m tight jobs (i.e., $d_j = t + 4$ for each job j of them) could be released at $t + 2$, forcing any c -competitive algorithm at $t + 2$ to have left not more work than $2m \cdot (c - 1)$ of the jobs due at $t + 3$. Also note that in an optimal schedule, all the jobs could be feasibly scheduled.

By the above observation, we get that A is not c -competitive if $w + 2m \cdot (2 - c) > 2cm$. Otherwise, we can upper bound the work done on the more loose jobs in the interval $[t, t + 2]$ by $2cm - w + 2m \cdot (c - 2) = 4m \cdot (c - 1) - w$, amounting to $4m \cdot (c - 1) - w + cm/2 = 4m \cdot (9c/8 - 1) - w$ in the interval $[t, t + 3]$. What remains (of the more loose jobs) at $t + 3$ is thus at least a total workload of $cm - 4m \cdot (9c/8 - 1) + w = 4m \cdot (1 - 7c/8) + w$, which is larger than w if and only if $c < 8/7$. \square

This allows us to prove the theorem.

Proof of Theorem 9. We assume there exists a c -competitive algorithm A for semi-online machine minimization. Obviously, the lemma can be applied for $J = \emptyset$ and $t = 0$ (note that $w = 0$). By re-applying the lemma to the resulting instance for $\lceil 3cm \rceil$ more times, we can be sure that we finally have $W_{A(J')}(t') > 3cm$ for some J' and t' (by the strict increase of this value with each iteration), which is a contradiction. \square

Now consider the online instead of semi-online case. Using our doubling technique as a black box (Theorem 1), we are able to derive a 12-competitive algorithm. In this subsection we will give an improved 9.38-competitive algorithm by utilizing the c -competitive algorithm of Devanur et al. [9], which is actually used for the special case when $p = 1$.

Theorem 10. *There exists a 9.38-competitive algorithm for the online machine minimization problem when $p_j = p$.*

As m is unknown, we use the density $\rho(t)$ as an estimation for the number of machines needed for the jobs released until time t , where

$$\rho(t) = p \cdot \max_{[a,b] \subseteq [0, d_{max}]} \frac{|\{j \in J(t) \mid [r_j, d_j] \subseteq [a, b]\}|}{b - a}.$$

It can be easily seen that for any interval $[a, b]$, $|\{j \in J(t) \mid r_j \leq t, [r_j, d_j] \subseteq [a, b]\}|$ represents the set of all the jobs that have been released until time t , and have to be finished within $[a, b]$. Thus $\rho(t)$ serves as a lower bound on $m(J(t))$. Using $\rho(t)$, we give the following algorithm (parameterized by α and c):

Algorithm: For α -tight jobs, run **EarlyFit**, i.e., (non-preemptively) schedule a job immediately when it is released. For α -loose jobs, run **EDF** with $c\rho(t)$ machines.

We show in the following that with some parameter c the algorithm gives a feasible schedule.

Lemma 6. *EarlyFit uses at most $\lfloor (1/\alpha + 1)\rho(t) \rfloor$ machines.*

Proof. Suppose the lemma is not true. Then at some time t the Early-Fit algorithm requires more than $\lfloor (\alpha + 1)\rho(t) \rfloor$ machines, which implies that by scheduling the α -tight jobs immediately, there are more than $(1/\alpha + 1)\rho(t)$ tight jobs overlapping at time t . Hence each of the α -tight job is released after $t - p$ and has a deadline at most $t + p/\alpha$. Therefore, it holds that

$$\rho(t) > \frac{(1/\alpha + 1) \cdot \rho(t) \cdot p}{p/\alpha + p} = \rho(t),$$

which is a contradiction. \square

From now on we only consider loose jobs and let $\rho'(t)$ be the density of these jobs. Let J be the set of all loose jobs and J' be the modified instance by replacing each loose job $j \in J$ with p unit size jobs with feasible time window $[r_j, d_j]$. We observe that instance J and J' share the same density $\rho'(t)$. Furthermore, there is an e -competitive algorithm for J' which uses $\lceil e\rho'(t) \rceil$ machines. Let $W_J(t)$ be the remaining workload of jobs by applying our algorithm to J with $c \geq e$ and $W_{J'}(t)$ be the remaining workload of jobs by applying the e -competitive algorithm to J' . We have the following load inequality:

Lemma 7. *For every time t it holds that*

$$W_J(t) \leq W_{J'}(t) + e \cdot \rho'(t) \cdot p.$$

Proof. We prove by induction on t . Obviously the lemma is true for $t = 0$ as $W_J(0) = W_{J'}(0)$. Suppose the lemma holds for $t \leq k$. We prove that $W_J(k+1) \leq W_{J'}(k+1) + e \cdot \rho'(k+1) \cdot p$.

Consider $w_J(k)$, i.e., the workload processed during $[k, k+1]$ by our algorithm. If $w_J(k) < \lceil e \cdot \rho'(k) \rceil \cdot p$, then there exists a free machines, implying that there are at most $e \cdot \rho'(k)$ jobs unfinished. Thus, we get $W_J(k+1) \leq W_{J'}(k+1) + e \cdot \rho'(k+1) \cdot p$. Otherwise, we get $w_J(k) \geq \lceil e \cdot \rho'(k) \rceil \cdot p$. Recall that the lemma is true for $t = k$. That means that we have $W_J(k) \leq W_{J'}(k) + e \cdot \rho'(k) \cdot p$. Since it holds that $W_J(k+1) = W_J(k) - w_J(k) \leq W_J(k) - \lceil e \cdot \rho'(k) \rceil \cdot p$, $W_{J'}(k+1) = W_{J'}(k) - w_{J'}(k) \geq W_{J'}(k) - \lceil e \cdot \rho'(k) \rceil \cdot p$ as well as $\rho'(k+1) \geq \rho'(k)$, simple calculations show that the lemma is true. \square

Using the above load inequality, we prove the following:

Lemma 8. *For $c \geq e \cdot (1 + \alpha)/(1 - \alpha)$, EDF is a $c\rho'(t)$ -competitive algorithm for online machine minimization if $p_j \equiv p$ and every job is α -loose.*

Proof. Among those instances EDF fails at, we pick one with the minimum number of jobs. It can be easily seen that in such an instance there is only one job missed, and this job has the largest deadline. Let j be the missed job. Then we know that until time $d_j - p/\alpha$, every job is released, implying that $\rho'(t)$ does not change after time $t = d_j - \lfloor p/\alpha \rfloor$. Let $\rho'(t) = \rho'_{max}$ for $t \geq d_j - \lfloor p/\alpha \rfloor$. Furthermore the fact that job j is missed implies that during $[d_j - \lfloor p/\alpha \rfloor, d_j]$, there must exist at least $\lfloor p/\alpha \rfloor - p + 1$ time units when every machine is busy. This means that

$$W_J(d_j - p/\alpha) > c\rho'_{max} \cdot (\lfloor p/\alpha \rfloor - p + 1) + p - 1.$$

On the other hand, the load inequality implies that

$$W_J(d_j - \lfloor p/\alpha \rfloor) \leq W_{J'}(d_j - \lfloor p/\alpha \rfloor) + e \cdot \rho'_{max} \cdot p \leq \lfloor p/\alpha \rfloor \cdot \lceil e\rho'_{max} \rceil + e \cdot \rho'_{max}.$$

Simple calculations show that choosing $c \geq e(1 + \alpha)/(1 - \alpha)$ yields a contradiction. \square

We can now prove the theorem.

Proof of Theorem 10. Combining the two algorithms for tight and loose jobs, we derive an $(e \cdot (1 + \alpha)/(1 - \alpha) + 1/\alpha + 1)$ -competitive algorithm. By taking $\alpha \approx 0.33$, we get a 9.38-competitive algorithm. \square

4.2 Non-preemptive scheduling

We consider non-preemptive semi-online machine minimization with equal processing times and also show a 4-competitive algorithm.

Let $P = \{0, p, 2p, \dots\}$ be the set of all the multiples of the integer p . For any job j , we let $Q_j = [r_j, d_j]$ be its feasible time window. We know that $Q_j \cap P \neq \emptyset$ since $d_j - r_j \geq p$. We now define job j to be a *critical* job if $|Q_j \cap P| = 1$, and a *non-critical* job if $|Q_j \cap P| \geq 2$.

When we say we round a non-critical job j , we round up its release date r_j to be its nearest value in P , and round down d_j to be its nearest value in P .

Algorithm: Open $2m$ machines to schedule the non-critical jobs on them. Upon the release of a new job, check if it is critical. If it is critical, schedule it using **EarlyFit**, i.e., schedule it immediately upon its release on a separate machine. Otherwise round it and add it to the set of all the non-critical jobs that have not been scheduled. At time τp and for any integer τ , pick $2m$ jobs from the set via EDF and schedule them on the dedicated machines.

We first prove the following lemma.

Lemma 9. *For every instance J , there exists a schedule that uses at most $2m$ machines such that every critical job is scheduled via **EarlyFit**, i.e., immediately, and every non-critical job is scheduled exactly in $[\tau p, (\tau + 1) \cdot p]$ for some integer τ .*

Proof. Consider the (offline) optimum schedule S for the instance J . We alter the schedule in the following way. Firstly, we schedule critical jobs immediately when they are released. Secondly, we move any non-critical job which is processed all the time during some $[\tau p - 1, \tau p + 1]$ (for τ integer) either completely into $[(\tau - 1) \cdot p, \tau p]$ or into $[\tau p, (\tau + 1) \cdot p]$. We claim that by doing so we increase the number of machines by at most m , which we show for each τ in the following.

Consider any critical job j' whose feasible time window intersects with P at point τp . We observe that in any feasible solution, and hence in the solution S , j' is being processed all the time during $[\tau p - 1, \tau p + 1]$. We let y_c be the number of these jobs. Similarly, we let y'_c be the number of those critical jobs that intersect with P at $(\tau + 1) \cdot p$.

Consider non-critical jobs. Let y_s be the number of non-critical jobs that are completely scheduled in $[\tau p, (\tau + 1) \cdot p]$ in S . Let y_l be the number of non-critical jobs whose feasible time windows contain $[\tau p, (\tau + 1) \cdot p]$ and which are processed within $[\tau p, \tau p + 1]$ in S but are not completely scheduled in $[\tau p, (\tau + 1) \cdot p]$. We know that

$$y_s + y_c + y_l \leq m(J).$$

Similarly, let y_r be the number of non-critical jobs whose feasible time windows contain $[\tau p, (\tau + 1) \cdot p]$ and which are processed within $[(\tau + 1) \cdot p, (\tau + 1) \cdot p - 1]$ in S but are not completely scheduled in $[\tau p, (\tau + 1) \cdot p]$. We have

$$y_s + y'_c + y_r \leq m(J).$$

Now after we alter the solution S , the maximum load during $[\tau p, (\tau + 1) \cdot p]$ is at most $y_c + y'_c + y_s + y_l + y_r \leq 2m(J)$. \square

We observe that, in a schedule that satisfies the above lemma, all non-critical jobs could actually be viewed as unit-size jobs. The following lemma can be proved via a simple exchange argument as in [13].

Lemma 10. *If there exists a feasible schedule for unit-size jobs that uses μ machines at time $[t, t + 1]$, then applying EDF to schedule these jobs with μ machines also returns a feasible solution.*

We are now ready to prove the theorem.

Theorem 11. *For equal processing times, there exists a 4-competitive algorithm for non-preemptive semi-online machine minimization.*

Proof. Consider the algorithm defined above. Using Lemma 9 and 10, we get that EDF for non-critical jobs requires at most $2m$ machines. Furthermore, Lemma 9 also implies that $2m$ machines suffice to schedule the critical jobs via EarlyFit. Thus, we need at most $4m$ machines to schedule the whole instance. \square

Using the Double as a black box (Theorem 1) for non-critical jobs, we directly obtain the following result.

Theorem 12. *For equal processing times, there exists a 10-competitive algorithm for non-preemptive online machine minimization.*

Remark. Lemma 9 actually implies a 4-approximation algorithm for the offline scheduling problem, in which the instance is known in advance and the goal is to minimize the number of machines so as to schedule each job in a feasible way.

Theorem 13. *There exists a 4-approximation algorithm for non-preemptive offline machine minimization when the processing times of all jobs are equal.*

Proof. Let m be the (unknown) optimum value. We schedule critical jobs immediately when they are released. This requires at most $2m$ machines by Lemma 9. On separate machines we schedule non-critical jobs, and after rounding they are viewed as unit-size jobs which could be scheduled feasibly on additional $2m$ machines according to Lemma 9. In [13] an optimum algorithm is presented for the offline problem of scheduling unit size jobs, and by applying this algorithm we directly get a feasible schedule for non-critical jobs on at most $2m$ machines. Thus, the overall approximation ratio is 3. \square

5 Uniform deadlines

Consider the special case of (semi-)online machine minimization with a uniform deadline, i.e., every job $j \in J$ has the same deadline $d_j = d$.

5.1 Preemptive Scheduling

First look at the special case of preemptive semi-online deadline scheduling in which every job $j \in J$ has the same deadline $d_j = d$. We prove that running LLF with m machines yields a feasible solution and thus LLF is 1-competitive.

Theorem 14. *On instances with a uniform deadline, LLF is 1-competitive for preemptive semi-online machine minimization.*

Proof. We prove via an exchange argument. Let S_{OPT} be the feasible offline schedule that uses m machines, and S_{LLF} be the schedule produced by LLF (and in S_{LLF} some jobs may miss their deadlines).

We show that we can inductively alter S_{OPT} to S_{LLF} and thereby maintain feasibility. To this end, suppose S_{OPT} and S_{LLF} coincide during $[0, t]$ for $t \geq 0$. Let j be any job which is chosen to be scheduled in S_{LLF} during $[t, t+1]$ but preempted in S_{OPT} . If there is still a free machine in S_{OPT} , we can simply use one of them to schedule j . Otherwise every machine is occupied and there exists some j' so that during $[t, t+1]$ job j' is processed in S_{OPT} but not in S_{LLF} . Recall that in S_{LLF} job j' is not processed in favor of job j . Thus their laxities satisfy $\ell_{j'}(t) \leq \ell_j(t)$, or equivalently, $p_{j'}(t) \geq p_j(t)$. Since in S_{OPT} job j' is chosen to be scheduled in $[t, t+1]$ while job j is preempted, it follows that during $[t+1, d]$ in S_{OPT} , there must exist some time t' such that job j is scheduled, while job j' is not scheduled (either preempted or finished). We now alter S_{OPT} by swapping the unit of j' processed during $[t, t+1]$ with the unit of j processed during $[t', t'+1]$. By doing so, the resulting schedule is still feasible. \square

We conclude this subsection with the following result obtained using Theorem 1.

Theorem 15. *On instances with a uniform deadline, there is a 4-competitive algorithm for preemptive online machine minimization.*

5.2 Non-preemptive Scheduling

Consider non-preemptive semi-online machine minimization with a uniform deadline.

Recall the definition of α -tight and α -loose jobs. Since $d_j = d$ in this special case, a job is α -tight if $p_j > \alpha(d - r_j)$, and α -loose otherwise. We give a 5.25-competitive algorithm.

Algorithm: We schedule (α)-tight and loose jobs separately. For tight jobs, we use **EarlyFit**, that is, schedule each job immediately when they are released. For loose jobs, we apply EDF with $m/(1 - \alpha)^2$ machines.

Remark. Since jobs have the same deadline, EDF actually does not distinguish among jobs and thus represents any busy algorithm.

Lemma 11. *EarlyFit is a $\lceil 1/\alpha \rceil$ -competitive algorithm for (semi-)online machine minimization.*

Proof. Let $k = \lceil 1/\alpha \rceil$ for simplicity. Then for any tight job j we have $p_j > 1/k \cdot (d - r_j)$. Consider the solution produced by **EarlyFit** on tight jobs and assume it uses m' machines. There exists some time t such that $w_{\text{EarlyFit}}(t) = m'$. Let S be this set of these m' jobs processed at t . We claim that

any $k+1$ jobs from S could not be put on the same machine in any feasible solution, yielding that $m \leq m'/k$.

To see why the claim is true, suppose that there exist $k+1$ jobs from S that can be scheduled on one machine, and let them be job 1 to job $k+1$ for simplicity. We know that $r_j \leq t$ for $1 \leq j \leq k+1$, i.e., we get $p_j > 1/k \cdot (d - r_j) \geq 1/k \cdot (d - t)$. Now consider the schedule where the $k+1$ jobs are scheduled on one machine, and assume w.l.o.g. that job 1 is scheduled at first. Recall that even if job 1 is scheduled immediately after its release, it will be processed during $[t, t+1]$. Thus job 2 to job $k+1$, which are scheduled after job 1, could not start before time $t+1$. Hence, they will be scheduled in $[t+1, d]$. However, $p_j > 1/k \cdot (d - t)$, which is a contradiction. \square

We can now prove the following theorem:

Theorem 16. *There is a 5.25-competitive algorithm for non-preemptive semi-online machine minimization with a uniform deadline.*

Proof. For loose jobs, we know that the scheduling problem with same deadline jobs is a special case of scheduling with agreeable deadlines jobs. Hence according to Lemma 3, EDF is a $1/(1-\alpha)^2$ -competitive semi-online algorithm. Combining the above lemma and setting $\alpha = 1/3$, we derive the 5.25-competitive algorithm. \square

Using Double as a black box (Theorem 1), we directly derive a 21-competitive algorithm for non-preemptive online machine minimization with a uniform deadline. However, since EarlyFit is already an online algorithm, we only need to apply the doubling technique to transform EDF into an online algorithm. This yields a $(4/(1-\alpha)^2 + \lceil 1/\alpha \rceil)$ -competitive algorithm. This performance guarantee is optimized for $\alpha = 1/4$, yielding the following result.

Theorem 17. *There is a $11\frac{1}{5}$ -competitive algorithm for non-preemptive online machine minimization with a uniform deadline.*

6 A Preemptive $\mathcal{O}(\log n)$ -competitive Algorithm

We give a preemptive $\mathcal{O}(\log n)$ -competitive semi-online algorithm and apply Double (Theorem 1). Assume for the remainder of this section that the optimal number of machines is known.

6.1 Description

We fix some parameter $\alpha \in (0, 1)$. At any time t , our algorithm maintains a partition of $J(t)$ into *safe* and *critical* jobs (at t), which we define as

$$L_t = L_{t-1} \cup \{j \in J(t) \mid p_j(t) \leq \alpha(d_j - t)\} \quad (L_{-1} = \emptyset) \text{ and } T_t = J(t) \setminus L_t, \text{ respectively.}$$

In other words, once a job gets α -loose at some time, it gets classified as safe job from then on and is only considered as critical if it has *always* been α -tight.

Our algorithm does the following. Once a job is classified as safe it is taken care of by EDF on separate machines. More formally, consider $L = \bigcup_t \{j_t \mid j \in L_t \setminus L_{t-1}\}$, where j_t is the *residue* of job j at time t , i.e., it is a job with processing time $p_j(t)$ and time window $[t, d_j]$. We run EDF on L using $m(L)/(1-\alpha)^2$ machines. The critical jobs (at time t) require a more sophisticated treatment on $\Theta(m \log |J(t)|)$ machines, which we describe below.

Consider the set of critical jobs T_t at time t . We partition T_t into $h_t \cdot \mu_t$ different subsets, each of which is processed by EDF on a separate machine. If t is not a release date, we simply take

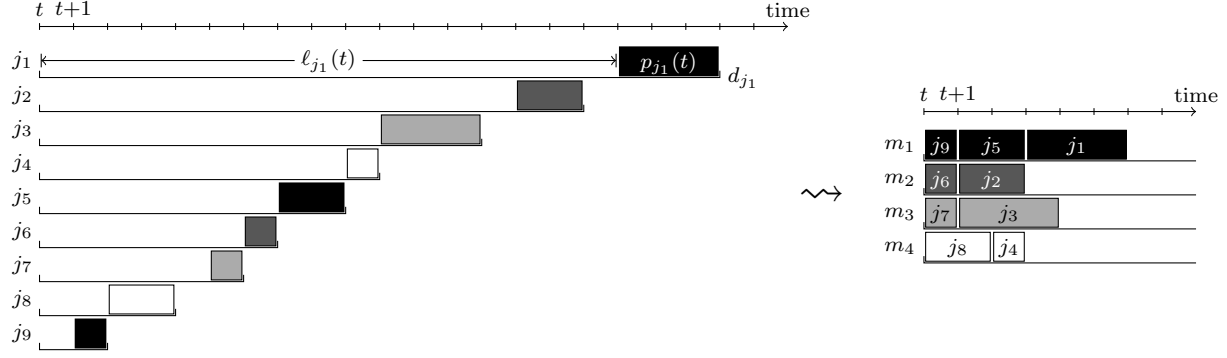


Figure 1: Visualization of jobs in set $S_i \subset T_t$ and their (temporal) distribution over machines. Left: Time window of each job with its processing volume pushed to the right. Right: resulting job-to-machine assignment. Assume α to be small enough that none of the jobs gets safe before completion as well as $\mu_t = 4$ (the machines are called m_1, \dots, m_4).

over the partition from $t - 1$ and remove all the jobs that have become safe since then. Otherwise, assume w.l.o.g. that $T_t = \{1, \dots, |T_t|\}$ with $d_j \geq d_{j+1}$, for all j , and first partition T_t into h_t many sets S_1, \dots, S_{h_t} . We consider the jobs in increasing order of indices and assign them to existing sets or create new ones, as follows. Let $a(i)$ be the job with the earliest deadline in an existing set S_i . Then a job j is added to an arbitrary set S_i such that the (remaining) length of the feasible time window of j is smaller than the laxity of j_i , i.e., $d_j - t \leq \ell_{a(i)}(t)$. If no such set exists, we create a new set and add j .

Consider an arbitrary S_i . By construction, we could feasibly schedule this set of jobs on a single machine using EDF. However, doing so may completely use up the laxity of some job in S_i , causing problems when new jobs are released in the future. We want to keep a constant fraction $\beta \in (0, 1)$ (independent of t) of the initial laxity $\ell_j(r_j)$ for each $j \in S_i$ by distributing them carefully over $\mu_t > 1$ machines. Consider again $S_i = \{j_1, \dots, j_{|S_i|}\}$ and assume an increasing order by deadlines. We further partition S_i into μ_t different subsets $S_i^1, \dots, S_i^{\mu_t}$ by setting $S_i^k = \{j_i \mid i \equiv k-1 \pmod{\mu_t}\}$ and schedule *each* of them on a separate machine via EDF. By the right choice of μ_t , this reduces the decrease in laxity sufficiently. The procedure is visualized in Figure 1.

We know from the preliminaries that EDF on L requires $m(L)/(1 - \alpha)^2$ machines. Hence the algorithm uses $\mathcal{O}(m(L) + \mu_t \cdot h_t)$ machines. In Subsection 6.4, we prove that we can choose $\mu_t = \mathcal{O}(\log |J(t)|)$ so that a constant β as above exists and that $h_t = \mathcal{O}(m)$, $m(L) = \mathcal{O}(m)$. The key to prove that lies in Subsection 6.3 where we characterize the relation between a decrease in the laxity and increase in the number of machines. Hereby, we use the compact representation of the optimum value presented in Subsection 6.2. In summary we need $\mathcal{O}(m) + \mathcal{O}(m) \cdot \mathcal{O}(\log |J(t)|) = \mathcal{O}(\log n) \cdot m$ machines.

6.2 Strong Density – A Compact Representation of the Optimum

The key to the design and analysis of online algorithms are good lower bounds on the optimum value. The offline problem is known to be solvable optimally in polynomial time, e.g., by solving a linear program or a maximum flow problem [11]. However, these techniques do not provide a quantification of the change in the schedule and the required number of machines when the job set changes. We derive an exact estimate of the optimum number of machines as an expression of workload that must be processed during some (not necessarily consecutive) intervals.

Let \mathcal{I} denote a set of k pairwise disjoint intervals $[a_h, b_h]$, $1 \leq h \leq k$. We define the *length* of \mathcal{I}

to be $|\mathcal{I}| = \sum_{1 \leq h \leq k} |b_h - a_h|$ and its *union* to be $\cup \mathcal{I} = \bigcup_{1 \leq h \leq k} [a_h, b_h]$.

Definition 1 (Strong density). *Let \mathcal{I} be as above and $Q_j = [r_j, d_j]$ be the feasible time window of job j . The contribution $\Gamma_j(\mathcal{I})$ of job j to \mathcal{I} is the least processing volume of j that must be scheduled within $\cup \mathcal{I}$ in any feasible schedule, i.e.,*

$$\Gamma_j(\mathcal{I}) = \max\{0, |(\cup \mathcal{I}) \cap Q_j| - \ell_j\} = \max\{0, |(\cup \mathcal{I}) \cap Q_j| - d_j + r_j + p_j\}.$$

The strong density ρ_s is defined as maximum total contribution of all jobs over all interval sets:

$$\rho_s = \max_{\mathcal{I} \subseteq \{[t, t+1] \mid 0 \leq t < d_{\max}\}} \frac{\sum_{j=1}^n \Gamma_j(\mathcal{I})}{|\mathcal{I}|}.$$

Our main result in this section is that $\lceil \rho_s \rceil$ is the exact value of an optimal solution. We give a combinatorial proof, but we also note that it follows from LP duality.

Theorem 18. $m = \lceil \rho_s \rceil$.

Proof. It is easy to see that $\lceil \rho_s \rceil$ is a lower bound on m since the volume of $\sum_{j=1}^n \Gamma_j(\mathcal{I})$ must be scheduled in $\cup \mathcal{I}$ of length $|\mathcal{I}|$. This yields a lower bound of $\lceil \sum_j \Gamma_j(\mathcal{I}) / |\mathcal{I}| \rceil$ for any \mathcal{I} .

It remains to show that $m \leq \lceil \rho_s \rceil$. Given a schedule, we denote by χ_h the number of time slots $[t, t+1]$ during which exactly h machines are occupied. For any feasible schedule that uses m machines, we obtain a vector $\chi = (\chi_m, \dots, \chi_0)$. We define a lexicographical order $<$ on these vectors: we have $\chi < \chi'$ if and only if there exists some h with $\chi_h < \chi'_h$ and $\chi_i = \chi'_i$, for all $i < h$. We pick the schedule whose corresponding vector is the smallest with respect to $<$.

We now construct a directed graph $G = (V, A)$ based on the schedule we pick. Let $V = \{v_0, \dots, v_{d_{\max}-1}\}$ where v_t represents the slot $[t, t+1]$. Let ϕ_t be the number of machines occupied during $[t, t+1]$. An arc from v_i to v_k exists iff $\phi_i \geq \phi_k$ and there exists some job j with $[i, i+1] \cup [k, k+1] \subseteq Q_j$ whereas j is processed in $[i, i+1]$ but not in $[k, k+1]$. Intuitively, an arc from v_i to v_k implies that one unit of the workload in $[i, i+1]$ could be carried to $[k, k+1]$.

We claim that in G there is no (directed) path which starts from some v_i with $\phi_i = m$, and ends at some v_ℓ with $\phi_\ell \leq m-2$. Suppose there exists such a path, say $(v_{i_1}, \dots, v_{i_\ell})$ with $\phi_{i_1} = m$ and $\phi_{i_\ell} \leq m-2$. Then we alter the schedule such that we move one unit of the workload from $[i_s, i_s+1]$ to $[i_{s+1}, i_{s+1}+1]$, for all $s < \ell$. By doing so, ϕ_{i_1} decreased and ϕ_{i_ℓ} increased by 1 each. By $\phi_{i_1} = m$ and $\phi_{i_\ell} \leq m-2$, we get that χ_m decreases by 1, contradicting the choice of the schedule.

Consider $V_m = \{v_i \mid \phi_i = m\}$ and let $P(V_m) = \{v_{i_1}, \dots, v_{i_\ell}\}$ be the set of vertices reachable from V_m via a directed path (trivially, $V_m \subseteq P(V_m)$). The above arguments imply that for any $v_i \in P(V_m)$, it holds that $\phi_i \geq m-1$. We claim that $\sum_j \Gamma_j(\mathcal{I}) = \sum_h \phi_{i_h}$ for $\mathcal{I} = \{[i_h, i_h+1] \mid 1 \leq h \leq \ell\}$. Note there is no arc going out from $P(V_m)$ (otherwise the endpoint of the arc is also reachable and would have been included in $P(V_m)$), i.e., we cannot move out any workload from $\cup \mathcal{I}$, meaning that the contribution of all jobs to \mathcal{I} is included in the current workload in $\cup \mathcal{I}$, i.e., $\sum_h \phi_{i_h}$. Thus,

$$\rho_s \geq \frac{\sum_h \phi_{i_h}}{|\mathcal{I}|} = \frac{\sum_h \phi_{i_h}}{|P(V_m)|}.$$

Notice that ϕ_{i_h} is either m or $m-1$, and among them there is at least one of value m , thus the right-hand side is strictly larger than $m-1$, which implies that $\lceil \rho_s \rceil \geq m$. \square

6.3 The Power of Laxity

Let J be an arbitrary instance and $\beta \in (0, 1)$ be rational. We analyze the influence that the decrease in the laxity of each job by a factor of β has on the optimal number of machines. To this end, let $J_\beta = \{j_\beta \mid j \in J\}$ be the modification of J where we increase the processing time of each job by a $(1 - \beta)$ -fraction of its initial laxity, i.e., $[r_{j_\beta}, d_{j_\beta}] = [r_j, d_j]$ and $p_{j_\beta} = p_j + (1 - \beta)(d_j - r_j - p_j)$ (or equivalently $\ell_{j_\beta} = \beta \cdot \ell_j$). Using a standard scaling argument, we may assume w.l.o.g. that for a fixed β all considered parameters are integral.

Obviously, we have $m(J_\beta) \geq m(J)$. In the following, we show $m(J_\beta) \leq \mathcal{O}(1/\beta) \cdot m(J)$ for instances J only consisting of sufficiently tight jobs.

Theorem 19. *For every J consisting only of $(1/2)$ -tight jobs and $\beta \in (0, 1)$, it holds that*

$$m(J_\beta) \leq \frac{4}{\beta} \cdot m(J).$$

In fact, we even show a slightly stronger but less natural result. More specifically, we split each job $j_\beta \in J_\beta$ into two subjobs: the *left part* of j_β , denoted j_β^{\leftarrow} , and the *right part* of j , denoted j_β^{\rightarrow} , where

$$\begin{aligned} [r_{j_\beta^{\leftarrow}}, d_{j_\beta^{\leftarrow}}] &= \left[r_j, r_j + \left(1 - \frac{\beta}{2}\right) \cdot \ell_j \right], p_{j_\beta^{\leftarrow}} = (1 - \beta) \cdot \ell_j \\ \text{and } [r_{j_\beta^{\rightarrow}}, d_{j_\beta^{\rightarrow}}] &= \left[r_j + \left(1 - \frac{\beta}{2}\right) \cdot \ell_j, d_j \right], p_{j_\beta^{\rightarrow}} = p_j, \end{aligned}$$

i.e., we split the former feasible time window at $r_j + (1 - \beta/2) \cdot \ell_j$ and distribute p_{j_β} over both emerging jobs in such a way that the right part has the processing time of the original job j . The sets containing all of these subjobs are $J_\beta^{\leftarrow} = \{j_\beta^{\leftarrow} \mid j \in J_\beta\}$ and $J_\beta^{\rightarrow} = \{j_\beta^{\rightarrow} \mid j \in J_\beta\}$, respectively. In the following, we show Theorem 19 even for $J_\beta^{\leftarrow} \cup J_\beta^{\rightarrow}$ instead of J_β . This implies, as can be easily seen, the statement for J_β .

In the proof, we define two more jobs based on j and $\gamma \in (0, 1)$, namely the γ -right-shortened and the γ -left-shortened variant of j , denoted $j^{\leftarrow\gamma}$ and $j^{\gamma\rightarrow}$ and contained in the sets $J^{\leftarrow\gamma}$ and $J^{\gamma\rightarrow}$, respectively. We define

$$\begin{aligned} [r_{j^{\leftarrow\gamma}}, d_{j^{\leftarrow\gamma}}] &= [r_j, d_j - (1 - \gamma) \cdot \ell_j], p_{j^{\leftarrow\gamma}} = p_j \\ \text{and } [r_{j^{\gamma\rightarrow}}, d_{j^{\gamma\rightarrow}}] &= [r_j + (1 - \gamma) \cdot \ell_j, d_j], p_{j^{\gamma\rightarrow}} = p_j, \end{aligned}$$

which means that we drop an amount of $(1 - \gamma) \cdot \ell_j$ from either side of the feasible time window while keeping the processing time. We then show the following lemma.

Lemma 12. *For every J and $\gamma \in (0, 1)$, it holds that*

$$m(J^{\leftarrow\gamma}) \leq \frac{1}{\gamma} \cdot m(J) \text{ and } m(J^{\gamma\rightarrow}) \leq \frac{1}{\gamma} \cdot m(J).$$

Proof. We representatively show the statement for $J^{\gamma\rightarrow}$; the argumentation for $J^{\leftarrow\gamma}$ is analogous. According to Theorem 18, there exists a set of k pairwise disjoint intervals \mathcal{I} such that

$$m(J^{\gamma\rightarrow}) = \sum_{j^{\gamma\rightarrow}} \frac{\Gamma_{j^{\gamma\rightarrow}}(\mathcal{I})}{|\mathcal{I}|}.$$

W.l.o.g. we assume that $\mathcal{I} = \{[a_h, b_h] \mid 1 \leq h \leq k\}$ with I_h where $a_h < a_{h+1}$ (hence, $b_h < b_{h+1}$), for all h . To derive the relationship between $m(J^{\gamma\rightarrow})$ and $m(J)$, we expand \mathcal{I} into a set of element-wise

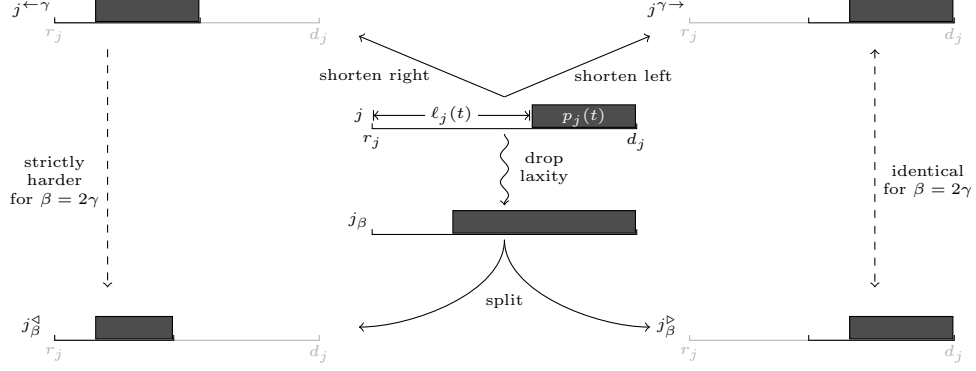


Figure 2: The different types of jobs defined for the proof of Theorem 19 and their relations. We let $\gamma = \beta/2 = 1/4$ and $p_j = 2 \cdot (d_j - r_j)/5$

larger intervals $\text{ex}(\mathcal{I})$ such that $|\mathcal{I}| \geq |\text{ex}(\mathcal{I})|/\gamma$. We show for each job j that $\Gamma_j(\text{ex}(\mathcal{I})) \geq \Gamma_{j^{\gamma \rightarrow}}(\mathcal{I})$ and thus the lemma follows.

The expanding works as follows. Given \mathcal{I} as above, we expand each of the intervals $[a_h, b_h]$ to $[a'_h, b_h]$ with $a'_h \leq a_h$ the following way. We start at the rightmost interval I_k and try to set $a'_k = b_k - (b_k - a_k)/\gamma$ for I'_k . If this would, however, produce an overlap between I'_k and I_{k-1} , we set $a'_k = b_{k-1}$, $\delta_k = b_{k-1} - (b_k - (b_k - a_k)/\gamma)$ and try to additionally expand I_{k-1} by δ_k instead. After that, we continue this procedure to the left but never expand an interval into negative time. More formally, we let $b_0 = \delta_{k+1} = 0$ and set for all h

$$a'_h = \max \left\{ b_{h-1}, b_h - \left(\frac{b_h - a_h}{\gamma} + \delta_{h-1} \right) \right\} \text{ and } \delta_h = \max \left\{ 0, b_{h-1} - \left(b_h - \left(\frac{b_h - a_h}{\gamma} + \delta_{h-1} \right) \right) \right\}.$$

The following two facts can be easily observed.

Observation 2. Consider \mathcal{I} as above. One of the following statements is true:

- (i) We have $\cup \text{ex}(\mathcal{I}) = [0, b_k]$ and $|\mathcal{I}| \geq |\text{ex}(\mathcal{I})|/\gamma$.
- (ii) We have $\cup \text{ex}(\mathcal{I}) \subseteq \left[a_1 - \frac{|\mathcal{I}|(1-\gamma)}{\gamma}, b_k \right]$ and $|\mathcal{I}| = |\text{ex}(\mathcal{I})|/\gamma$.

Observation 3. If $\cup \mathcal{I} \subseteq \cup \mathcal{I}'$, then $\cup \text{ex}(\mathcal{I}) \subseteq \cup \text{ex}(\mathcal{I}')$.

We now consider any job $j^{\gamma \rightarrow}$ with $\Gamma_{j^{\gamma \rightarrow}}(\mathcal{I}) > 0$ and claim that $\Gamma_j(\text{ex}(\mathcal{I})) \geq \Gamma_{j^{\gamma \rightarrow}}(\mathcal{I})$. First plug in the definition of contribution. We get

$$\begin{aligned} \Gamma_j(\text{ex}(\mathcal{I})) &= \max\{0, |(\cup \text{ex}(\mathcal{I})) \cap Q_j| - \ell_j\} \\ \text{and } \Gamma_{j^{\gamma \rightarrow}}(\mathcal{I}) &= \max\{0, |(\cup \mathcal{I}) \cap Q_{j^{\gamma \rightarrow}}| - \ell_{j'}\} = \max\{0, |(\cup \mathcal{I}) \cap Q_{j^{\gamma \rightarrow}}| - \gamma \ell_j\}. \end{aligned}$$

It thus suffices to prove that $|(\cup \text{ex}(\mathcal{I})) \cap Q_j| - (1 - \gamma) \cdot \ell_j \geq |(\cup \mathcal{I}) \cap Q_{j^{\gamma \rightarrow}}|$.

Let $\mathcal{I}' = \{I \cap Q_{j^{\gamma \rightarrow}} \mid I \in \mathcal{I}\}$ where we let a' be the leftmost point of it and b' be the rightmost one. By Observation 3, it follows that $\cup \text{ex}(\mathcal{I}') \subseteq \cup \text{ex}(\mathcal{I})$, i.e., we can restrict to proving

$$|(\cup \text{ex}(\mathcal{I}')) \cap Q_j| - (1 - \gamma) \cdot \ell_j \geq |\mathcal{I}'|.$$

According to Observation 2, there are two possibilities.

Case 1. We have $\cup \text{ex}(\mathcal{I}') = [0, b']$. Given the fact that $Q_j = Q_{j^{\gamma \rightarrow}} \cup [r_j, r_j + (1 - \gamma) \cdot \ell_j]$ and $r_j < b'$, it follows that $|(\cup \text{ex}(\mathcal{I}')) \cap Q_j| \geq |\mathcal{I}'| + (1 - \gamma) \cdot \ell_j$.

Case 2. It holds that $\text{ex}(\mathcal{I}')$ takes up a length of $|\mathcal{I}'|/\gamma$ in the interval

$$[a'', b'] = \left[a' - \frac{|\mathcal{I}'|(1-\gamma)}{\gamma}, b' \right].$$

Hence for any $x \geq 0$, it takes up a length of at least $|\mathcal{I}'|/\gamma - x$ in the interval $[a'' + x, b']$. Using $\Gamma_{j \rightarrow}(\mathcal{I}) = \Gamma_{j \rightarrow}(\mathcal{I}') > 0$, we get $|\mathcal{I}'| > \ell_{j \rightarrow} = \gamma \ell_j$. Thus, if we let $x = (|\mathcal{I}'| - \gamma \ell_j)(1-\gamma)/\gamma$, we get that $\text{ex}(\mathcal{I}')$ takes up a length of at least $|\mathcal{I}'| + (1-\gamma) \cdot \ell_j$ in the interval $[a'' + x, b'] = [a' + (1-\gamma) \cdot \ell_j, b']$. Since $[a' + (1-\gamma) \cdot \ell_j, b'] \subseteq Q_j$, we have $|(\cup \text{ex}(\mathcal{I}')) \cap Q_j| \geq |\mathcal{I}'| + (1-\gamma) \cdot \ell_j$, which completes the proof. \square

Proof of Theorem 19. We show that $m(J_\beta^\triangleleft \cup J_\beta^\triangleright) \leq 4 \cdot m(J)/\beta$ or, more specifically, that $m(J_\beta^\triangleleft) = m(J_\beta^\triangleright) = 2 \cdot m(J)/\beta$.

For the first part, compare the two instances J_β^\triangleleft and $J^{\leftarrow \beta/2}$. We observe that

$$[r_{j_\beta^\triangleleft}, d_{j_\beta^\triangleleft}] \subseteq [r_{j^{\leftarrow \beta/2}}, d_{j^{\leftarrow \beta/2}}] \quad \text{and} \quad \ell_{j_\beta^\triangleleft} = \ell_{j^{\leftarrow \beta/2}},$$

for all $j \in J$, where we use for the first part that j is $(1/2)$ -tight. This implies that a schedule of $J^{\leftarrow \beta/2}$ on m' machines can be easily transformed into a schedule of J_β^\triangleleft on m' machines. Hence, we get that $m(J_\beta^\triangleleft) \leq m(J^{\leftarrow \beta/2})$ and the claim follows by Lemma 12.

For the second part, we simply observe that $J_\beta^\triangleright = J^{\beta/2 \rightarrow}$ and apply Lemma 12 again. \square

6.4 Analysis

For the sake of simplicity, we choose $\alpha \in (0, 1/2]$ such that $1/\alpha$ is integral. We now analyze the performance of our algorithm. The goal of this subsection is proving the following theorem.

Theorem 20. *Our algorithm is an $\mathcal{O}(\log n)$ -competitive algorithm for the machine minimization problem.*

Recall that we apply EDF to schedule the safe jobs $L = \bigcup_t \{j_t \mid j \in L_t \setminus L_{t-1}\}$. Using Theorem 2, $m(L)/(1-\alpha)^2 = \mathcal{O}(m(L))$ machines are suffice to feasibly schedule all the safe jobs. Meanwhile at any time $[t, t+1]$, we partition T_t into h_t subsets S_1, \dots, S_{h_t} and schedule μ_t jobs of each subset on a separate set of machines.

We first prove that, if we choose a certain $\mu_t = \mathcal{O}(\log |J(t)|)$, the laxity of each job in T_t never drops below a constant fraction of its original laxity. This implies that our algorithm never misses a critical job. Hence our algorithm does not miss any job.

Lemma 13. *We can choose $\mu_t = \mathcal{O}(\log |J(t)|)$ such that at any time t and for any $j \in T_t$, $\ell_j(t) \geq \beta \ell_j$ where $\beta = 2 - \pi^2/6 \in (0, 1)$.*

Proof. Let $t' \leq t$ be an arbitrary time when the partition of $T_{t'}$ into the sets $S_1, \dots, S_{h_{t'}}$ is recomputed. Let $S_i \ni j$ and w.l.o.g. $S_i = \{1, \dots, |S_i|\}$ where $d_{j'} \geq d_{j'+1}$, for all j' . By the way the partition is constructed, we further have $d_{j'+1} - t \leq \ell_{j'}(t')$. Now let $j^* = j + \mu_{t'}$ and notice that, unless the partition is recomputed, the execution of j is only blocked by other jobs for at most $d_{j^*} - t'$ (if j^* does not exist, j is immediately executed). Since every job in $T_{t'}$ is α -tight at t' , this however means that the execution of j is only delayed by $(1-\alpha)^{\mu_{t'}} \cdot \ell_j(t') \leq (1-\alpha)^{\mu_{t'}} \cdot \ell_j$. Note that we can choose $\mu_{t'} = \mathcal{O}(\log |J(t')|)$ in such a way that $(1-\alpha)^{\mu_{t'}} \cdot \ell_j \leq \ell_j/|J(t')|^2$. To get the

total time j is at most delayed by, we have to sum over all such t' . Recall that at least one job is released at any such time t' , hence $|J(t')| \geq |J(t' - 1)| + 1$ and we have:

$$\ell_j(t) \geq \ell_j - \sum_{i=2}^{|J(t)|} \frac{\ell_j}{i^2} \geq \ell_j - (\pi^2/6 - 1) \cdot \ell_j = (2 - \pi^2/6) \cdot \ell_j.$$

□

Before proving the main theorem, we need an additional bound on h_t . To this end, let $\hat{T}_t = \{j_t \mid j \in T_t\}$ be the set of residues of T_t at t , i.e., the jobs in T_t with remaining processing times and time windows.

Lemma 14. *At all times t , it holds that*

$$h_t \leq 1 + \left(2 + \frac{2}{\alpha}\right) \cdot m(\hat{T}_t).$$

W.l.o.g., we let $\hat{T}_t = \{1, \dots, |\hat{T}_t|\}$ where $d_j \geq d_{j+1}$, for all j . We first prove the following auxiliary lemma:

Lemma 15. *For any j and $j' = j + 2m(\hat{T}_t)/\alpha$, we have $d_{j'} - t \leq (d_j - t)/2$.*

Proof. Suppose there exists some j such that $d_{j+h} - t > (d_j - t)/2$ holds for any h with $0 \leq h \leq 2m(\hat{T}_t)/\alpha$. Given that every job in T_t is α -tight at t , we get $p_{j+h} > \alpha/2 \cdot (d_j - t)$. Thus, the total workload that has to be finished in the interval $[t, d_j]$ is larger than $\alpha/2 \cdot (d_j - t) \cdot 2m(\hat{T}_t)/\alpha = (d_j - t) \cdot m(\hat{T}_t)$, which contradicts the fact that $m(\hat{T}_t)$ machines are enough to accommodate \hat{T}_t . □

Proof of Lemma 14. Recall the construction of the different S_i and consider the state of S_1, \dots, S_{h_t-1} when S_{h_t} is opened. Then there exists some job j which could not be added into any of the subset S_1 to S_{h_t-1} . As before, we let λ_i be the earliest-deadline job from S_i . Obviously job λ_i is added to S_i before we consider job j , $\lambda_i < j$. According to Lemma 15, there are at least $h_t - 1 - 2 \cdot m(\hat{T}_t)/\alpha$ jobs among $\lambda_1, \dots, \lambda_{h_t-1}$ whose feasible time window has a length at least $2(d_j - t)$, and has a remaining laxity no more than $d_j - t$. Thus if we consider the interval $[t, t + 2(d_j - t)]$, each of the $h_t - 1 - 2 \cdot m(\hat{T}_t)/\alpha$ contribute at least a processing time of $d_j - t$, implying that the workload that has to be finished in this interval is at least $(h_t - 1 - 2 \cdot m(\hat{T}_t)/\alpha)(d_j - t)$. On the other hand, as $m(\hat{T}_t)$ machines are enough to accommodate all the jobs, the workload processed during $[t, t + 2(d_j - t)]$ is at most $m(\hat{T}_t) \cdot 2(d_j - t)$. Hence $(h_t - 1 - 2 \cdot m(\hat{T}_t)/\alpha)(d_j - t) \leq m(\hat{T}_t) \cdot 2(d_j - t)$, which implies the lemma. □

We are ready to prove the main theorem of this section:

Proof of Theorem 20. Recall that the number of machines used by our algorithm is $\mathcal{O}(m(L) + h_t \cdot \mu_t)$. Using Lemma 13, it only remains to bound $m(L)$ and h_t . We set $\beta = 2 - \pi^2/6 \in (0, 1)$.

We first show $h_t \leq 1 + (2 + 2/\alpha) \cdot 4m/\beta = \mathcal{O}(m)$ for all t . According to Lemma 14, it suffices to prove that $m(\hat{T}_t) \leq 4m/\beta$. By Lemma 13, we know that for any job j , we have $\ell_j(t) \geq \beta \ell_j$. Consider the instance J_β as defined in Subsection 6.3. Any feasible schedule of J_β implies a feasible schedule of \hat{T}_t , i.e., we get $m(\hat{T}_t) \leq m(J_\beta)$. Theorem 19 implies that $m(J_\beta) \leq 4m/\beta$ and the claim follows.

Moreover, we show $m(L) \leq 4m/\beta = \mathcal{O}(m)$. Consider any job $j_t \in L$. If job j is released at time t , then $\ell_j(t) = \ell_j$. Otherwise we have $\ell_j(t - 1) \geq \beta \ell_j$ according to Lemma 13. As j is α -tight at $t - 1$ and becomes α -loose at t , it follows that j is processed in $[t - 1, t]$, implying that $\ell_j(t) = \ell_j(t - 1) \geq \beta \ell_j$. Thus it holds that $\ell_j(t) \geq \beta \ell_j$. As a consequence, any feasible schedule of J_β implies a feasible schedule of L , implying that $m(L) \leq m(J_\beta) \leq 4m/\beta$ by Theorem 19. □

7 Lower Bounds

7.1 Lower bound for LLF

Phillips et al. [17] showed that the competitive ratio of LLF for (semi-)online machine minimization is not constant. We extend their results by showing that LLF requires $\Omega(n^{1/3})$ machines.

Theorem 21. *There exists an instance of n jobs with a feasible schedule on m machines for which LLF requires $\Omega(n^{1/3})$ machines.*

Proof. Let m be even. For any $\hat{c} > 1$, we give an instance at which LLF using $\hat{c}m = cm/2$ machines fails. Towards this, consider the integer sequence (x_0, x_1, x_2, \dots) with x_0 large enough and $x_r = x_0 \sum_{i=r+2}^{\infty} (1/c)^i$, for all $r \geq 1$. Let $G_t(t, r)$ be the set of $m/2$ identical (*more tight*) jobs with feasible time window $[t, t + x_0 \sum_{i=r}^{\infty} (1/c)^i]$ and laxity $x_0 \sum_{i=r+1}^{\infty} (1/c)^i$. Moreover, let $G_l(t, x_r)$ be the set of $cm/2$ identical (*more loose*) jobs with feasible time window $[t, t + cx_r]$ and processing time x_r . We construct the instance in k rounds (where k is yet to be specified) as follows.

- *Round 1.* From time 0 to time x_0 , we release $G_t(0, 0)$ and $G_l(t, x_1)$ for $t = icx_1$ where $i = 0, 1, \dots, x_0/(cx_1) - 1$. It can be easily verified that LLF will always preempt jobs in $G_t(0, 0)$ in favor of jobs in $G_l(t, x_1)$. Thus at time x_0 , each job in $G_t(0, 0)$ is preempted for exactly $x_1 \cdot x_0/(cx_1) = x_0/c$ time units, i.e., by time x_0 there are still $m/2$ jobs in $G_t(0, x_0)$, each having a remaining processing time of x_0/c to be scheduled within $[x_0, x_0 + x_0 \sum_{i=1}^{\infty} (1/c)^i]$. However, the optimum solution could finish all the jobs released so far by time x_0 .
- *Round $r > 1$.* Carry on the above procedure. Suppose at time $x_0 \sum_{i=0}^{r-2} (1/c)^i$ the optimum solution could finish all the jobs released so far while for LLF there are still $(r-1) \cdot m/2$ jobs, each having a remaining processing time x_0/c^{r-1} to be scheduled within $[x_0 \sum_{i=0}^{r-2} (1/c)^i, x_0 \sum_{i=0}^{\infty} (1/c)^i]$. Then from time $x_0 \sum_{i=0}^{r-2} (1/c)^i$ to time $x_0 \sum_{i=0}^{r-1} (1/c)^i$, we release $G_t(x_0 \sum_{i=0}^{r-2} (1/c)^i, r-1)$ and $G_l(t, x_r)$ for $t = x_0 \sum_{i=0}^{r-2} (1/c)^i + j \cdot cx_r$ where $j = 0, 1, \dots, x_0/(c^r x_r)$. It can be easily seen that at time $x_0 \sum_{i=0}^{r-2} (1/c)^i$ there are in total $(r) \cdot m/2$ jobs having a processing time x_0/c^{r-1} to be scheduled within $[x_0 \sum_{i=0}^{r-2} (1/c)^i, x_0 \sum_{i=0}^{\infty} (1/c)^i]$. Each of these jobs will be preempted in favor of jobs in $G_l(t, x_r)$. Thus until time $x_0 \sum_{i=0}^{r-2} (1/c)^i$, it is preempted for $x_r \cdot x_0/(c^r x_r) = x_0/c^r$ time units. This implies that there are $rm/2$ jobs, each having a processing time x_0/c^r to be scheduled within $[x_0 \sum_{i=0}^{r-1} (1/c)^i, x_0 \sum_{i=0}^{\infty} (1/c)^i]$. However, the optimum solution could finish all the jobs released so far by time x_r .

We estimate the number of jobs released at each round by $m + cm \cdot x_0/(c^{k+2} x_{k+2}) = c(c-1)m + m = \mathcal{O}(c^2 m) = \mathcal{O}(\hat{c}^2 m)$. Thus, until round k we release in total $\mathcal{O}(k \hat{c}^2 m)$ jobs. It is easy to see that LLF requires $\mathcal{O}(km)$ machines for the remaining jobs and $\hat{c}m$ machines will be insufficient if we choose some $k = \mathcal{O}(\hat{c})$. Thus it suffices to choose some $n = \mathcal{O}(\hat{c}^3)$ for the lower bound, i.e., LLF requires $\Omega(cm) = \Omega(n^{1/3})$ many machines. \square

7.2 Lower bound for Deadline-Ordered Algorithms

Consider deadline-ordered algorithms that schedule jobs using only the relative order of jobs instead of their actual values. EDF belongs to this class of algorithms. Lam and To [15] derived a lower bound on the speed that is necessary to feasibly schedule jobs on m machines. We modify their instance and argumentation to give the following lower bound on the number of unit-speed machines.

Theorem 22. *There are instances of n jobs with a feasible schedule on m machines for which any deadline-ordered algorithm requires at least $n - 1$ machines.*

Proof. We define a collection of instances with identical job sets that all can be feasibly scheduled on m machines. The only difference in the instances are the deadlines but the relative order is again the same. Thus, a deadline-ordered algorithm cannot distinguish the instances and must produce the same schedule for all of them. We show that this property enforces that (nearly) each job must run on its own machine.

For each $k \in \{1, 2, \dots, n - m\}$ we define the job set J_k as follows:

$$\begin{aligned} \text{For } 1 \leq j \leq m : \quad & r_j = 0, p_j = 1, d_j = \left(\frac{m}{m-1}\right)^k =: \bar{d}_k. \\ \text{For } m+1 \leq j \leq m+k : \quad & r_j = 0, p_j = \left(\frac{m}{m-1}\right)^{j-m}, d_j = \bar{d}_k. \\ \text{For } m+k+1 \leq j \leq n : \quad & r_j = 0, p_j = \left(\frac{m}{m-1}\right)^{j-m}, d_j = \left(\frac{m}{m-1}\right)^{n-m} =: \bar{d}. \end{aligned}$$

For every instance J_k , $k \in \{1, 2, \dots, n - m\}$, there is a feasible solution on m machines. Schedule the first $m+k$ jobs in order of their indices one after the other filling a machine up to the deadline \bar{d}_k before opening the next machine. Since every job has a processing time less than \bar{d}_k , no job will run simultaneously on more than one machine. The total processing volume is

$$\sum_{j=1}^{m+k} p_j = m + \sum_{\ell=1}^k \left(\frac{m}{m-1}\right)^\ell = m + \frac{\left(\frac{m}{m-1}\right)^{k+1} - 1}{\left(\frac{m}{m-1}\right) - 1} - 1 = m \left(\frac{m}{m-1}\right)^k = m \cdot \bar{d}_k,$$

and thus, all jobs are feasibly scheduled on m machines. The remaining jobs are scheduled by the same procedure in the interval $[\bar{d}_k, \bar{d}]$. With a similar argumentation as above, this is a feasible schedule. In particular, the total processing volume is

$$\sum_{j=m+k+1}^n p_j = \sum_{\ell=1}^{n-m-k} \left(\frac{m}{m-1}\right)^{\ell+k} = m \left(\left(\frac{m}{m-1}\right)^{n-m} - \left(\frac{m}{m-1}\right)^k \right) = m \cdot (\bar{d} - \bar{d}_k).$$

We now show that any fixed schedule must use $n - 1$ machines to guarantee a feasible solution for all instances J_k . With the argumentation above this implies the lower bound for any deadline-ordered algorithm.

W.l.o.g. we may assume that the order of job indices is the order of deadlines. We could enforce this explicitly by adding a small value, but omit it for the sake of presentation.

Any deadline-ordered solution must complete the first $m+k$ jobs by time \bar{d}_k to be feasible for J_k for $k \in \{1, \dots, n - m\}$. Since in any instance J_k the largest among those jobs has processing time $p_{m+k} = \bar{d}_k$, each job with index $m+k$ must receive its exclusive machine. That means, we need $n - m$ machines for jobs $\{m+1, m+2, \dots, n\}$. The remaining m jobs with unit processing time must finish in instance J_1 by time $\bar{d}_1 = \frac{m}{m-1}$ which requires $m - 1$ machines. In total, a deadline-ordered algorithm needs $n - 1$ machines whereas optimal solutions with m machines exist. \square

Notice that this very fatal lower bound is achieved already by instances in which all jobs are released at the same time. Thus, it is not the lack of information about further job arrivals but the lack of precise deadline information that ruins the performance.

8 Concluding Remarks

We contribute new algorithmic results for a fundamental online resource minimization problem. We give the first constant competitive algorithms for a major subclass in which jobs have agreeable deadlines, and also improve on upper bounds for the general problem. It remains as a major open question if the general preemptive problem admits a constant competitive ratio. We believe that some of our techniques may be useful for the closely related online throughput maximization problem, where we relax the requirement that all jobs must be scheduled (and fix a number of machines). It would also be interesting to quantify the tradeoff between the number of machines and the guaranteed throughput.

Acknowledgment

We thank Naveen Garg for discussions on deadline-ordered algorithms and his contribution to the lower bound in Theorem 22. Moreover, we thank Benjamin Müller for interesting discussions at an early stage of the project. As part of his Master’s thesis he contributed to Thm. 1 and part of Thm. 10.

References

- [1] S. Albers, F. Müller, and S. Schmelzer. Speed scaling on parallel processors. *Algorithmica*, 68(2):404–425, 2014.
- [2] S. Anand, N. Garg, and N. Megow. Meeting deadlines: How much speed suffices? In *Proceedings of the 38th International Colloquium on Automata, Languages and Programming (ICALP 2011)*, volume 6755 of *LNCS*, pages 232–243. Springer, 2011.
- [3] E. Angel, E. Bampis, and V. Chau. Low complexity scheduling algorithms minimizing the energy for tasks with agreeable deadlines. *Discrete Applied Mathematics*, 175:1–10, 2014.
- [4] N. Bansal, T. Kimbrel, and K. Pruhs. Speed scaling to manage energy and temperature. *J. ACM*, 54(1), 2007.
- [5] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.
- [6] M. Chrobak and C. Kenyon-Mathieu. SIGACT news online algorithms column 10: Competitiveness via doubling. *SIGACT News*, 37(4):115–126, 2006.
- [7] J. Chuzhoy, S. Guha, S. Khanna, and J. Naor. Machine minimization for scheduling jobs with interval constraints. In *Proceedings of the 45th Symposium on Foundations of Computer Science (FOCS 2004)*, pages 81–90. IEEE Computer Society, 2004.
- [8] M. Cieliebak, T. Erlebach, F. Hennecke, B. Weber, and P. Widmayer. Scheduling with release times and deadlines on a minimum number of machines. In *Proceedings of the 3rd IFIP International Conference on Theoretical Computer Science (TCS 2004)*, pages 217–230, 2004.
- [9] N. R. Devanur, K. Makarychev, D. Panigrahi, and G. Yaroslavtsev. Online algorithms for machine minimization. *CoRR*, abs/1403.0486, 2014.

- [10] M. R. Garey and D. S. Johnson. Two-processor scheduling with start-times and deadlines. *SIAM J. Comput.*, 6(3):416–426, 1977.
- [11] W. A. Horn. Some simple scheduling algorithms. *Naval Research Logistics Quarterly*, 21(1):177–185, 1974.
- [12] L. Jeż, F. Li, J. Sethuraman, and C. Stein. Online scheduling of packets with agreeable deadlines. *ACM Trans. Algorithms*, 9(1):5:1–5:11, 2012.
- [13] M.-J. Kao, J.-J. Chen, I. Rutter, and D. Wagner. Competitive design and analysis for machine-minimizing job scheduling problem. In *Proc. of the 23rd International Symposium (ISAAC 2012)*, volume 7676 of *LNCS*, pages 75–84, 2012.
- [14] A. J. Kleywegt, V. S. Nori, M. W. P. Savelsbergh, and C. A. Tovey. Online resource minimization. In R. E. Tarjan and T. Warnow, editors, *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 576–585. ACM/SIAM, 1999.
- [15] T. W. Lam and K.-K. To. Trade-offs between speed and processor in hard-deadline scheduling. In *Proc. of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 623–632, 1999.
- [16] C. A. Phillips, C. Stein, E. Torng, and J. Wein. Optimal time-critical scheduling via resource augmentation. In *Proceedings of the 29th Annual ACM Symposium on the Theory of Computing (STOC)*, pages 140–149. ACM, 1997.
- [17] C. A. Phillips, C. Stein, E. Torng, and J. Wein. Optimal time-critical scheduling via resource augmentation. *Algorithmica*, 32(2):163–200, 2002.
- [18] B. Saha. Renting a cloud. In *Proc. of the IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2013)*, volume 24 of *LIPICs*, pages 437–448, 2013.
- [19] Y. Shi and D. Ye. Online bin packing with arbitrary release times. *Theor. Comput. Sci.*, 390(1):110–119, 2008.
- [20] G. Yu and G. Zhang. Scheduling with a minimum number of machines. *Oper. Res. Lett.*, 37(2):97–101, 2009.